MSc THESIS

Testing of PUF based secure ICs

Gijs Roelofs

Abstract

Secure systems based on Physical Unclonable Functions (PUFs) are gaining popularity due to their hardware intrinsic security. Achieving high quality in such a system requires high quality manufacturing tests; hence, a good controllability and observability of the internal nodes of the system are required. However, this makes it easier for hackers to get access to internal secret information. Therefore, providing a high quality test while making it very hard/impossible to hack the system is a major challenge.

This thesis targets a secure test approach for Fuzzy Extractors (FEs), which constitutes the main component of PUF systems. Two novel Built-in self-test (BIST) methods are proposed: daisy chaining and parallel. Both methods provide a higher security compared to existing methods, due to the fact that they are both scan-chain free and do not require any external connections. The daisy chaining test method reuses existing blocks in the FE for pattern generation and output compression to minimize the area overhead; the method realizes a fault coverage of 95% at the cost of 50k clock cycles and an area overhead of only 2.2%. The parallel test method tests individual and simultaneously each of the Fuzzy Extractor blocks with a Random Test Pattern Generator (RTPG), minimizing the overall test time; the test method realizes a fault coverage of 95% at the cost of 8k clock cycles and an area overhead of 18.6%. The fault coverage can be increased at the cost of either test time or area overhead. The methods presented are generic and are applicable to any Fuzzy Extractor.

Faculty of Electrical Engineering, Mathematics and Computer Science
Testing of PUF based secure ICs

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

EMBEDDED SYSTEMS

by

Gijs Roelofs
born in Amsterdam, The Netherlands

Computer Engineering
Department of Electrical Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Testing of PUF based secure ICs

by Gijs Roelofs

Abstract

Secure systems based on Physical Unclonable Functions (PUFs) are gaining popularity due to their hardware intrinsic security. Achieving high quality in such a system requires high quality manufacturing tests; hence, a good controllability and observability of the internal nodes of the system are required. However, this makes it easier for hackers to get access to internal secret information. Therefore, providing a high quality test while making it very hard/impossible to hack the system is a major challenge.

This thesis targets a secure test approach for Fuzzy Extractors (FEs), which constitutes the main component of PUF systems. Two novel Built-in self-test (BIST) methods are proposed; daisy chaining and parallel. Both methods provide a higher security compared to existing methods, due to the fact that they are both scan-chain free and do not require any external connections. The daisy chaining test method reuses existing blocks in the FE for pattern generation and output compression to minimize the area overhead; the method realizes a fault coverage of 95% at the cost of 50k clock cycles and an area overhead of only 2.2%. The parallel test method tests individual and simultaneously each of the Fuzzy Extractor blocks with a Random Test Pattern Generator (RTPG), minimizing the overall test time; the test method realizes a fault coverage of 95% at the cost of 8k clock cycles and an area overhead of 18.6%. The fault coverage can be increased at the cost of either test time or area overhead. The methods presented are generic and are applicable to any Fuzzy Extractor.

Laboratory : Computer Engineering
Codenumber  : CE-MS-2013-01
Committee Members :

Advisor: Ir. Geert-Jan Schrijen, Intrinsic ID, The Netherlands
Chairperson: Dr.Ir. Said Hamdioui, CE, TU Delft, The Netherlands
Member: Dr.Ir. Rene van Leuken, CAS, TU Delft, The Netherlands
Member: Dr.Ir. Stephen Wong, CE, TU Delft, The Netherlands
Member: Ir. Mafalda Cortez, CE, TU Delft, The Netherlands
Contents

List of Figures viii
Acknowledgements ix

1 Introduction 1
  1.1 Information Security 1
  1.2 Physical Unclonable Functions 3
  1.3 Testing PUF based systems 3
  1.4 Contribution of the thesis 4
  1.5 Outline of the thesis 4

2 PUF based Secure Systems 7
  2.1 Overview of a Secure System 7
  2.2 Physical Unclonable Functions 8
  2.3 Explicitly-Introduced Randomness PUFs 8
    2.3.1 Optical PUFs 8
    2.3.2 Coating PUFs 9
  2.4 Intrinsic Randomness PUFs 10
    2.4.1 Ring Oscillator PUFs 10
    2.4.2 SRAM PUFs 11
    2.4.3 Butterfly PUFs 11
  2.5 PUF Reproducibility and Uniqueness 12
  2.6 PUF-based secure systems 13
    2.6.1 Enrollment phase 14
    2.6.2 Key Reconstruction phase 14
  2.7 Fuzzy Extractor 16
    2.7.1 Blocks of a Fuzzy Extractor 17
    2.7.2 Functioning of a Fuzzy Extractor 18
  2.8 Reliability requirements for a Fuzzy Extractor 20
    2.8.1 Test requirements 20
    2.8.2 Security requirements 21
  2.9 Summary 21

3 State of the art in secure IC testing 23
  3.1 Test concept 23
  3.2 Functional vs. structural testing 24
    3.2.1 Functional testing 24
    3.2.2 Structural testing 25
    3.2.3 Combination 27
3.3 ATE vs BIST ......................................................... 27
  3.3.1 External (ATE) testing ........................................ 28
  3.3.2 Internal (BIST) testing ....................................... 28
  3.3.3 Combination .................................................. 29
  3.3.4 Comparison from a security point of view .................... 29
3.4 Protecting Secure ICs after manufacturing ............................ 29
  3.4.1 Protection at the I/O ports .................................... 30
  3.4.2 Protection within the system ................................. 32
3.5 Summary ......................................................... 35

4 Built-in self-test ........................................... 37
  4.1 Built-in self test concept ........................................ 37
    4.1.1 Test Pattern Generation .................................... 38
    4.1.2 Compaction ................................................ 38
  4.2 Test Pattern Generation ......................................... 39
    4.2.1 Linear feedback shift register ........................... 40
    4.2.2 Cellular automata ......................................... 42
    4.2.3 Generalized LFSR ......................................... 42
    4.2.4 LFSR + phase shifter ..................................... 43
    4.2.5 Cryptographic cores ...................................... 44
    4.2.6 Summary ................................................. 45
  4.3 Reconfigurable LFSR for improving fault coverage ............... 45
    4.3.1 Output reconfiguration .................................... 46
    4.3.2 State reconfiguration ..................................... 48
    4.3.3 Structure reconfiguration ................................ 51
    4.3.4 Combination ............................................. 52
  4.4 Summary ......................................................... 53

5 Test development for a Fuzzy Extractor ........................... 55
  5.1 Fuzzy Extractor functional model ................................ 55
  5.2 Communication within the Fuzzy Extractor ......................... 56
  5.3 Golay Encoder ................................................. 58
  5.4 Repetition Encoder ............................................ 58
  5.5 Repetition Decoder ............................................... 59
  5.6 Golay Decoder .................................................. 60
  5.7 Hash Function .................................................. 61
  5.8 Remaining components ........................................... 62
  5.9 Test methods and procedure ..................................... 62
    5.9.1 Daisy-chaining ............................................ 62
    5.9.2 Parallel .................................................. 64
  5.10 Summary ........................................................ 65

iv
6 Results

### 6.1 Experiments
- 6.1.1 Daisy-chaining
- 6.1.2 Parallel

### 6.2 Test results of daisy-chaining

### 6.3 Test results of parallel
- 6.3.1 Golay Encoder
- 6.3.2 Repetition Encoder
- 6.3.3 Repetition Decoder
- 6.3.4 Golay Decoder
- 6.3.5 Universal Hash
- 6.3.6 Impact of reconfigurable LFSR/RTPG parameters

### 6.4 Comparison of the test methods

### 6.5 Recommendations

### 6.6 Summary

7 Conclusion & Future work

### 7.1 Contribution

### 7.2 Future work

Bibliography
## List of Figures

1.1 Examples of secure systems ........................................... 2
2.1 Generic secure system .................................................. 7
2.2 Optical PUF ............................................................... 9
2.3 Coating PUF ............................................................. 9
2.4 Block diagram of a Ring Oscillator PUF ............................... 10
2.5 Crossed couple inverters of an SRAM cell .......................... 11
2.6 Block diagram of a Butterfly PUF: Cross-coupled latches ...... 11
2.7 Metrics of a PUF response ............................................. 13
2.8 PUF-based secure system ............................................. 14
2.9 Enrollment phase ....................................................... 15
2.10 Reconstruction phase ................................................ 15
2.11 Fuzzy extractor ....................................................... 16
2.12 Hash function ......................................................... 18
2.13 Enrollment phase ..................................................... 19
2.14 Reconstruction phase ................................................ 20
3.1 A washing machine ..................................................... 23
3.2 Schematic overview of the testing process .......................... 24
3.3 Test approach classification .......................................... 25
3.4 Test point insertion .................................................... 26
3.5 Overview of a scan chain ............................................. 27
3.6 Overview of a BIST implementation ................................. 28
3.7 Overview of test port protecting methods .......................... 30
3.8 Challenge response based test system ............................... 31
3.9 Removing sensitive information when switching to system mode 31
3.10 Scan enable protection ............................................... 33
3.11 Spy flipflops in the scan chain .................................... 33
3.12 Scan chain scrambling ............................................... 34
3.13 Inverters in the scan chain ......................................... 35
4.1 Components of a Built-in self-test ................................. 37
4.2 An external lfsr ......................................................... 40
4.3 Cellular automata ....................................................... 42
4.4 Genaralised LFSR ..................................................... 43
4.5 Randomness comparison of GLFSR ................................. 43
4.6 LFSR plus phase shifter ............................................. 44
4.7 Crypto module of AES or DES ..................................... 44
4.8 A reconfigurable LFSR ............................................... 46
4.9 Output methods ......................................................... 47
4.10 Bit-flipping/fixing .................................................... 47
4.11 Reconfigurable Interconnection Network ......................... 48
Acknowledgements

Foremost I would like to express my most sincere thanks to my supervisor from Delft University of Technology (TU Delft): Said Hamdioui for providing me with this master thesis project. It was a challenging subject, but proved to be a great learning experience. In addition, I would like to thank my day to day supervisor, Mafalda Cortez. Her advises helped me to guide this thesis project in the right direction and she proved to be of great help by reviewing my thesis.

From intrinsicID I would like to thank Geert-Jan Schrijen, for cooperating with the TU-Delft to make this thesis project possible and providing the financial support. In addition I would like to thank Peter Simons, Marten van Hulst and Erik van der Sluis. Their hardware knowledge about PUF systems proved to be of great help, developing the test solution proposed in this thesis.

From Laboratoire d’Informatique, de Robotique et de Microelectronique de Montpellier (LIRMM), I would like to thank Giorgio di Natale. His input during the weekly meetings and technical support through email, proved to be of great help in developing the concept of Functional BIST. In addition I would like to thank him for providing me with the LIFTING fault simulator.

Lastly but definitely not the least, I would like to thank my parents and family for supporting me both financially and non-financially. Without their support this thesis would not have been possible.

Gijs Roelofs
Delft, The Netherlands
October 17, 2013
Introduction

This chapter introduces the basic concepts of a Secure System and the motivation of testing; it also highlights the main contributions of the work and provides the outline of this thesis. Section 1.1 briefly introduces Information Security and the vulnerabilities of existing secure systems, focuses on cryptographic key storage techniques and concludes that Physical Uncloneable Functions provide a novel and secure method for on-chip key storage. Section 1.2 discusses the concept of Physical Uncloneable Functions, including their advantages, limitations and challenges. Section 1.3 elaborates the testing challenges of PUF-based systems. Section 1.4 highlights the main contributions of this thesis and Section 1.5 contains the thesis outline.

1.1 Information Security

Since the beginning of the digital revolution in circa 1980 the amount of data stored and processed has increased significantly. However, some of this data contains sensitive or vulnerable information, such as bank accounts or identification information, that has to be protected. This protecting of information is called Information Security (IS). IS is defined in the code of the United States of America [1] as "means of protecting information and information systems from unauthorized access, use, disclosure, disruption, modification, or destruction in order to provide:

- **Integrity**: guarding against improper information modification or destruction, and includes ensuring information nonrepudiation and authenticity;
- **Confidentiality**: preserving authorized restrictions on access and disclosure, including means for protecting personal privacy and proprietary information;
- **Availability**: ensuring timely and reliable access to and use of information.

A system that assures that data meets the above requirements is called a secure system. The success of a secure system is determined by the asymmetry between the effort required to protect the information and the effort to break its protection [2]. All protection methods of a secure system can thus be broken, given enough time and resources [3, 4, 5, 6]. For most devices this is not of big concern: they only have to be secured for at least the time until the device is replaced by a new model. A recent example can be seen in the console market. The security of the Microsoft Xbox 360 was broken within weeks after launch [7] while that of the Sony PlayStation 3 was broken after four years [8]. Therefore, the Sony PlayStation 3 can be considered a more secure device.
CHAPTER 1. INTRODUCTION

than the Microsoft Xbox 360. As the consoles have an expected lifetime of around ten years [9, 10], it can be expected that due to the lower piracy on the PlayStation 3, Sony is able to make more profit with the console than Microsoft. It can be assumed that the investments of Sony, spends on security, is higher as well. A trade-off between the investment in security and the profit has thus to be made.

Some examples of a secure system commonly found in a household are depicted in Figure 1.1. To assure the confidentiality of the data stored in the system, mathematical algorithms are used, which are called encryption algorithms. These algorithms are publicly known: The security is based on the concept of secret keys that are securely stored somewhere in the system [5, 11, 12]. The security of a system is compromised when an attacker gains access to the secret key or paraphrasing Kerckhoff’s axiom: ”A system should be secure even if everything about the system, except the key, is public knowledge” [4, 13, 14]. For storing the key two different methods exist:

1. Non-volatile memory (NVM): is a type of memory which can store data even when the power of the system is off. Examples of NVMs are: Flash, Read Only Memory (ROM) and Battery backed memory. From a security standpoint, NVMs have one weakness in common: the permanent presence of the key in the device in a digital form. Flash memory has several well-known attacking methods to read out their content, even when the device is off. A list of attacking methods can be found in the literature [5, 15, 16, 17].

2. Physical Unclonable Functions (PUF): are functions embedded on-chip, of which a key can be generated. The key generation is only performed when needed by the encryption algorithm. When not needed, the key is removed from the system. This prevents attackers from retrieving the key when the device is idle or turned off. PUFs do require extra components for generating and processing the key. A PUF-based system thus uses more area than traditional storage methods and require more power.
Despite their disadvantages, PUFs are considered more secure than non-volatile memory. Therefore, this thesis focuses on PUF-based systems.

## 1.2 Physical Unclonable Functions

Every hardware device contains properties that are unique for the device. These are caused by inherent deep-submicron process variation during the manufacturing process. By using these variations a Device Specific Fingerprint can be generated. The fingerprints are able to uniquely identify a device, just like an iris scan does with humans. A system that is capable of producing the fingerprints is called a Physical Unclonable Function (PUF). These words can be defined as follows:

- **Physical**: A PUF system is physically embedded in the device. It should not be feasible to separate, modify or tamper the PUF, without being noticed by the system.

- **Unclonable**: PUFs are based on random physical properties of a device. As these are intrinsically created during the manufacturing process, PUFs are considered extremely difficult to either physically reproduce (hardware clone) or to model.

- **Function**: In mathematics, functions consist of three components: input, output and a relation that maps the input on the output. PUF systems have a similar behavior: a PUF input is challenged and an output response is created according to a certain mapping.

The fingerprint can be used in a secure device to generate a cryptographic key. However, PUF fingerprints are noisy. This means that some bits of the fingerprint may not be the same in between successive readout operations, hence a circuit is necessary to correct these bits. This circuit is called a Fuzzy Extractor [3]. To guarantee the proper functioning of a device, it needs to be tested after manufacturing. The testing of a PUF-based system is explained next.

## 1.3 Testing PUF based systems

The manufacturing of modern devices requires a large number of processing steps [18]. As a consequence, not all manufactured devices are functioning correctly. To separate the faulty devices from the good devices, all devices need to be tested. Currently two approaches to test devices exist:

- **External**: this is the most popular method; it modifies the circuit by adding extra wires and components to enhance the testing process [19]. Via these modifications an external test machine can reach the internals of the device and scan the circuit for defects. However, for secure devices this approach cannot be used as an attacker can use these features to gain access to the cryptographic key or partially encrypted data.
CHAPTER 1. INTRODUCTION

- **Internal**: this method is based on the integration of the test facilities on the device itself. As no extra access points to the internals of the device are created, this is considered a more secure method compared to external methods. However, this method suffers from a large area overhead, long test time or the possibility of not detecting all faults in the circuit.

Providing a high quality test approach, while satisfying the security requirements is the topic of this work.

1.4 Contribution of the thesis

This thesis proposes two new methods for testing a Fuzzy Extractor (FE); daisy-chaining and parallel. Daisy-chaining reuses components already available in the FE, which results in a very low area overhead. Parallel introduces a new test infrastructure, thereby achieving a very fast test time. Both methods achieve high fault coverage and high security compared to existing solutions. The main contribution of the thesis are:

- The design of two efficient test methods for a Fuzzy Extractor that provide a very high quality test, with very high security; daisy-chaining and parallel. Daisy-chaining is based on reusing existing components, parallel on a new test infrastructure.
- Demonstration that existing components can be reused as efficient methods for pattern generation and output compression.
- A new classification of test method optimizations. By using this new classification accurate predictions can be made on which method can be used best, with its expected trade-offs.
- Fuzzy Extractor experiment that prove the efficiency of the two proposed methods; daisy-chaining and parallel. Both methods result in a high fault coverage of 95%. Daisy-chaining requires 50k clock cycles and has an area overhead of just 2.2%. Parallel tests the blocks at the same time and provides a fast test time of 8k clock cycles with an area overhead overhead of 18.6%.
- A paper and a journal based on the results of this thesis; the paper is submitted to the *Design, Automation & Test in Europe 2014* conference, the journal will be submitted to the *IEEE Transaction on Very Large Scale Integration Systems* journal.

1.5 Outline of the thesis

The remainder of this thesis is organized as follows: Chapter 2 gives an introduction into secure systems based on PUFs and explains how such a system functions. Chapter 3 discusses the state-of-the-art in testing of a secure IC. Chapter 4 discusses Built-in self-test in which a device contains the logic to test itself. Chapter 5 discusses the development
of a test solution for a Fuzzy Extractor. Chapter 6 describes the implementation and results and Chapter 7 concludes this thesis and discusses the future work.
One of the main vulnerabilities in today’s secure system is the storage of the cryptographic key. This chapter discusses an alternative method for key storage known as Physical Unclonable Functions. Section 2.1 explains the behavior of a generic Secure System. Section 2.2 discusses the concept of Physical Unclonable Functions and its two main categories. Section 2.3 explains the category of Explicitly-Introduced Randomness PUFs and Section 2.4 the category of Intrinsic Randomness PUFs. Section 2.5 introduces the concept of Uniqueness and Reproducibility, which are two main metrics to evaluate a PUF. Section 2.6 explains the key generating mechanisms in PUF based secure systems. Section 2.7 describes a Fuzzy Extractor which is the main component in PUF based systems and Section 2.8 discusses the requirements a test solution has to fulfill in order to be used for a PUF-based secure system. Section 2.9 summarizes the main ideas discussed in this chapter.

2.1 Overview of a Secure System

Modern day households contain an increasing number of Secure Systems. Examples are Wi-Fi access points, digital access keys, bank systems, etc. An overview of a generic secure system is depicted in Figure 2.1. Assuming a user wants to store data in an insecure medium, e.g., the cloud, the information contained in the stored data has to be protected against unauthorized access. This protection can be achieved by a method called encryption. Encryption is the method of encoding data in such a way that eavesdroppers or hackers cannot read it, but that authorized parties can. Two well-known encryption standards are the Advanced Encryption Standard (AES) and the Data Encryption Standard (DES). A detailed explanation of these standards can be found in [20, 21]. To encrypt the file, the user has to provide the system with a secret token that is used by the encryption algorithm to make sure that only the users that knows the secret token can decrypt the files. This unique token is called the key. The classic approach stores the cryptographic key permanently in the system. However, this approach
is highly vulnerable to physical attacks [5]. Physical Unclonable Functions (PUFs) are an alternative to permanently cryptographic key storage, efficiently generating the cryptographic key, from sub-micron characteristics that are unique for each device. PUFs are discussed next.

### 2.2 Physical Unclonable Functions

Physical Unclonable Functions (PUFs) are physical structures embedded, typically, in Integrated Circuits (IC) [3]. PUFs have unique characteristics per device due to inherent deep-submicron process variations during manufacturing. Instead of storing a key, PUFs can be used to generate a unique key for every device. PUFs are considered Unclonable as they are extremely difficult to exactly copy (hardware clone) or to model in software. To clone a PUF, an attacker would have to gather information, with very high accuracy, of all the variations inside the silicon. Since semiconductor devices contain a very large number of random variations, this becomes an infeasible task. PUFs can be classified in two categories [3]:

1. **Explicitly-Introduced Randomness PUFs**: are those that introduce the randomness artificially during manufacturing. The advantage of this method, is that the PUF response can be optimized for the intended system. The disadvantage is that manufacturing cost increase due to the cost of extra materials or processing steps.

2. **Intrinsic Randomness PUFs**: are PUFs that are inherently created during the manufacturing process. The advantage of intrinsic PUFs is that they do not require any additional steps in the manufacturing process. The disadvantage of this PUF type is that the response may not be as random as required and it is more sensitive to external variations, e.g., temperature [3].

These two categories are explained in more depth in the following sections.

### 2.3 Explicitly-Introduced Randomness PUFs

Explicitly-Introduced Randomness PUFs are PUFs artificially created during the manufacturing process. Due to this, the manufacturer has some control over the characteristics of the PUF response, for example by means of optimizing the material. There are currently two well-known Explicitly-Introduced Randomness PUFs [3]: the Optical PUF and the coating PUF. These are discussed in detail in the following subsections.

#### 2.3.1 Optical PUFs

Optical PUFs are created by doping a transparent material, for example glass, with light scattering particles [2, 22]. In addition, a laser is added to the system. An overview
2.3. EXPLICITLY-INTRODUCED RANDOMNESS PUFs

Figure 2.2: Optical PUF [23]

is depicted in Figure 2.2. The challenge of the PUF is performed by highlighting the transparent material with a laser. The response is a unique speckle pattern, due to the randomly distributed light scattering particles within the material. This pattern is extremely hard to predict and nearly impossible to reproduce. Different responses of the same material can be accomplished by changing the angle of incidence of the laser on the material.

The advantages of Optical PUFs are a good level of difference among devices (easy to distinguish devices from each other) and good resistance to external variations, e.g. temperature. The main disadvantage is that the response is not direct available in a digital form. In [2] Pappu proposes to add a Charged-Couple Device (CCD) to convert the light emitted by the speckle pattern into a digital format. However, this costs area and requires extra processing steps.

2.3.2 Coating PUFs

Coating PUFs are another type of Explicitly-Introduced Randomness PUF. A Coating PUF is created by spraying the top of a device with an opaque material which is doped with dielectric particles [24]. An overview can be seen in Figure 2.3. The dielectric particles form a randomly distributed capacitance on top of the device. In order to be used as a PUF, this capacitance need to be evaluated. The reading operation is performed by measuring the capacitance through an array of sensor located in the top metal layer of the device. To guarantee sufficient differences in the randomness in between devices, care has to be taken that the size of the particles does not exceed the distance in between the sensor parts [24].

Figure 2.3: Coating PUF [24]
The advantage of Coating PUFs is the protection against invasive attacks. Non-invasive attacks are attacks where a device is analyzed without modifying its structure. Coating PUFs are resistant to this due to the fact that the opaque coating does not allow any light to pass through and hence it is not possible to inspect the circuit underneath the coating layer. Another advantage is that if this layer is altered, the PUF response changes accordingly and the intrusion can be detected. The main drawback of Coating PUFs is that the value they produce is analog. To readout this value analog-to-digital converters have to be used, which cost area, require extra processing steps and create noise [24].

2.4 Intrinsic Randomness PUFs

Intrinsic PUFs derive their name from the fact of being inherently created during the manufacturing process. An advantage of this category of PUFs is that they provide a response directly in the binary language. Three well known Intrinsic Randomness PUFs are the Ring Oscillator PUF, the SRAM PUF and the Butterfly PUF. They are explained in the coming subsections.

2.4.1 Ring Oscillator PUFs

Ring Oscillator PUFs were presented for the first time in [25]. An overview of such a system is depicted in Figure 2.4. The concept is as follows: due to intrinsic variation in the manufacturing process, small differences in delays between components exist. These differences can be measured by implementing a self-oscillating circuit [26]. An example is a circuit with an uneven number of inverters placed in a loop. By measuring the number of iterations in a predefined time the oscillating-frequency of this circuit can be determined, which is then considered the PUF response. One concern might be the dependency of the circuit on the temperature. In [26] the authors claim that by placing several loops in a chip and measuring the difference instead of the absolute value, the PUF becomes more resistant to environmental changes.

The advantage of Ring Oscillator PUFs is the one common to all Intrinsic Randomness PUF: the possibility of being integrated in an IC design with no extra steps required. However, they suffer from being not as random as other types of PUFs [3].
2.4. INTRINSIC RANDOMNESS PUFS

2.4.2 SRAM PUFs

An SRAM memory consists of an array of memory cells. The core of a memory cell consists of two cross coupled inverters. An overview is depicted in Figure 2.5.

Figure 2.5: Crossed couple inverters of an SRAM cell [23]

Although the inverters are designed symmetrically, due to process variation, small deviations exist. The asymmetry between the inverters can be used to generate a PUF response. When the SRAM is powered-up, the two inverters begin a race. The strongest inverter wins and imposes a value on the cell [27]. This value is called the start-up value. The set of start-up values of the memory cells are the PUF response.

The advantage of this PUF is that it is already available in most designs as almost all devices contain some form of SRAM. The disadvantage is that SRAM-PUFs are more sensitive to environmental variations than some other PUFs types [3].

2.4.3 Butterfly PUFs

Butterfly PUFs (BPUF) were proposed as a solution for FPGAs. Not all FPGAs contain an integrated memory and some manufacturers initialize their memory [28]. In Figure 2.6 an example of a BPUF can be seen. Although implemented differently, the concept

Figure 2.6: Block diagram of a Butterfly PUF: Cross-coupled latches [29]
is similar to that of SRAM PUFs: an cross-coupled system is formed in which the components race against each other until a stability point is reached. It is composed of two cross-coupled latches. In the starting state of the system the system is reset and there inputs are supplied with an equal input value. Once the reset is removed the elements start a race where the strongest component imposes his value on the cell.

The advantage of BPUFs is that they can be placed all over the device [3]. If placed next to a large logic block, they are much harder to detect than the regular structured SRAM memories. The disadvantage is that they use more area than an SRAM PUF and are sensitive to environmental variations.

All PUFs have their advantages and disadvantages. The choice of a certain PUF should thus be made depending on the application. Nevertheless, for the great majority of applications the SRAM PUF is favorite. This is due to the fact that SRAMs can be found on almost any device and that the integration does not require any additional processing steps. Although the methods described in this thesis are generic for every type of PUF, the main focus is on SRAM PUFs.

2.5 PUF Reproducibility and Uniqueness

PUF’s fingerprints, regardless of their type, have two main metrics to measure their quality to be used in a PUF system; they are:

- **Reproducibility**: Any challenge of the same PUF should provide, ideally, an equal response. However, when a device is powered-up several times and each of the corresponding fingerprints is compared, small differences occur. These differences are undesirable and are known as *noise*.

- **Uniqueness**: any PUF fingerprint should, ideally, be on average 50% different when compared with any other PUF fingerprint. Due to this characteristic, it is possible to uniquely identify each device. However, the bits of a PUF fingerprint are slightly dependent on each other. These dependencies are undesirable and are known as *correlation*.

Figure 2.7 illustrates the two metrics.
2.6 PUF-BASED SECURE SYSTEMS

Figure 2.7: Metrics of a PUF response

Figure 2.7(a) shows the patterns of the same device when powered two times; it can clearly seen that the patterns are almost the same except for the dots in the (red) circles. However, Figure 2.7(b) depicts the patterns of two different devices; it can be observed that the patterns are on the average 50 percent different.

Secure Systems require a stable PUF response, i.e., low noise, to guarantee its proper operation. In addition the correlation has to be removed. This can be achieved by introducing a component called a Fuzzy extractor. A secure system based on this component is the topic of the next section.

2.6 PUF-based secure systems

A PUF-based secure system performs two main operations called the Enrollment phase and the Key reconstruction phase. During the Enrollment phase the cryptographic key is defined and error correcting codes are added for robustness. During the Key Reconstruction phase errors are corrected and the correlation is removed. To perform these operations in a PUF system four components are required as depicted in the example of Figure 2.8.

The components are:

1. PUF (e.g., SRAM): generates random and unique patterns for every device.

2. Fuzzy Extractor: corrects the noise and removes the correlation.

3. Non-volatile memory (NVM): stores data used by the Fuzzy Extractor to correct the noise. This data is called helper data.

4. Stream cypher: encrypts the data provided by the user (optional and only necessary for this example).

The Enrollment and Key reconstruction phase are discussed in detail in the coming sections.
2.6.1 Enrollment phase

The Enrollment phase is responsible for defining the cryptographic key. The Enrollment phase is only performed once, in a safe environment, as this phase defines the key in the system. Figure 2.9 depicts an example architecture with the four main steps necessary to perform the Enrollment phase. It is described as follows:

1. The Enrollment phase is activated by setting the enroll pin.

2. A random is provided at the id_in port and processed by the Fuzzy Extractor. Inside the Fuzzy Extractor error correcting bits are added to make the data robust.

3. The PUF is challenged (e.g. SRAM power-up) and the device response is read (puf_in).

4. The input data (cryptographic key + error correction) is encoded with the PUF response and the resulting helper data is stored in a NVM.

2.6.2 Key Reconstruction phase

The Key Reconstruction phase is responsible for reconstructing the key whenever it is needed by the system. It can be performed in an insecure environment. Figure 2.10 depicts the six steps of the procedure that can be explained as follows:

1. The enroll pin is deactivated.
2.6. PUF-BASED SECURE SYSTEMS

2. The helper data stored in the NVM is read by the Fuzzy Extractor.

3. The PUF is challenged and the device specific pattern is generated. This pattern can be slightly different than the pattern obtained during Enrollment. The original PUF pattern is recovered by means of error correction bits stored in the helper data.
4. The cryptographic key is compacted by hashing the error corrected PUF pattern and available for use, for example, by a stream cipher. The hash removes the correlation in the data.

5. The user data is read by the stream cipher.

6. The encrypted user data is available at the output of the stream cipher.

The main component of the system, and the focus of this thesis is the Fuzzy Extractor. It is explained in the next section.

2.7 Fuzzy Extractor

As discussed in section 2.5, a PUF should be reproducible and unique. However, a real PUF response contains some noise and is slightly correlated, hence, PUF-based secure systems require extra processing to correct these issues. This component responsible for this processing is called a Fuzzy Extractor (FE) [30]. For this thesis we base our analysis on a reference FE design that has been developed within the Unique project [31]. An example design of such a component is depicted in Figure 2.11. It comprises

![Fuzzy Extractor diagram](image)

Figure 2.11: Fuzzy extractor

five blocks: Golay Encoder, repetition Encoder, Repetition Decode, Golay Decoder and a Hash Function. The five blocks have two main purposes:

1. **Error correction**: to overcome the noise problem, error correction is used. Several error correction methodologies exist and they all have a trade-off between the


2.7. FUZZY EXTRACTOR

number of bits they can correct and the overhead of bits they require. To make the optimal correction capability given a certain amount of bits, several error correcting schemes have to be combined.

2. Privacy amplification: PUF responses are not truly random. This means that there is a small correlation within the bits of the response. As the helper data is correlated with the PUF response, it is not truly random either. In order to derive a key with full entropy the corrected PUF response is hashed.

The blocks of the Fuzzy Extractor play a critical role in the enrollment and reconstruction phase. They are described in more detail next.

2.7.1 Blocks of a Fuzzy Extractor

The Fuzzy Extractor evaluated in this thesis consists of a Repetition encoding scheme implemented in the blocks: Repetition Encoder and Decoder and a Golay encoding scheme implemented in the blocks: Golay Encoder and Decoder. The hash block implements a universal hash. These are discussed next.

Repetition Encoder and Decoder

A repetition code is an error correction method that repeats every bit of the message N times. It is assumed that the receiver knows the value of N, hence, it can use majority voting to reconstruct the original message. If the number of errors in the set of N bits is less then $\lfloor N/2 \rfloor$ the message is successfully corrected. For example, consider the message $m = 101$ and a repetition factor of $N = 3$. The encoded message $m'$ is then: 111000111. If an error occurs in $m'$, for example: 101000111, the original message can be reconstructed by using majority voting. Due to the fact that majority voting is used, $N$ cannot be an even number. This error correction is a simple method, but comes at a price of a higher overhead.

Golay Encoder and Decoder

A more efficient error correction method is the Golay code. There are several types of Golay Codes. An overview can be found in [32]. The most popular Golay code is the extended binary Golay code which is described as Golay [24, 12, 8]. It maps 12 bits input vectors onto 24 bits output vectors in such a way that they differs in at least 8 coordinates. As a consequence, during the decoding up to 3 errors can be corrected. A detailed explanation can be found in [33]. An advantage of Golay codes is that it is more efficient then Repetition codes. A disadvantage of Golay codes is that it requires the number of inputs to be exactly 12.

Hash Function

A hash functions maps a given input of arbitrary length into a fixed size output. A hash
function implements a one way function: It is easy to calculate the value of the output from the input. However, for a given output value, it is almost impossible to determine the input value. This property can be used to increase the randomness of the output bits. An example is depicted in Figure 2.12. Figure 2.12(a) depicts the hash block:

![Hash block diagram](image)

Figure 2.12: Hash function

It handles up to 1024 input numbers that are mapped on 4 output numbers. Figure 2.12(b) depicts the mapping of the numbers. Determining from the input the output is easy, e.g., input 6 maps on output 2. However, determining from the output the input is impossible. Take as an example output 3; It is not possible to determine whether the input leading to this value was 15 or 66.

### 2.7.2 Functioning of a Fuzzy Extractor

In the this section, the functioning of a Fuzzy Extractor is explained according to the phases discussed before: the Enrollment phase and the Key Reconstruction phase.

**Enrollment phase**

During the Enrollment phase the Fuzzy Extractor is responsible for two tasks: (i) Adding error correcting bits to make the data robust against noise and (ii) encoding the cryptographic key with the PUF response. An overview of the process can be seen in Figure 2.13 and is explained next.

1. In the first stage a random seed is provided at id_in.
2. Error correction codes are applied to the cryptographic key in the form of Golay encoding.
3. The second encoding scheme applied is in the form of Repetition codes.
4. The PUF is challenged. The data is then XORed to form the helper data.
5. The helper data is stored in a non-volatile memory (e.g., flash).
2.7. FUZZY EXTRACTOR

In the Enrollment phase the cryptographic key is enhanced with error correction codes, encoded with the PUF response and stored in the non-volatile memory. The Reconstruction phase is responsible for the reverse procedure. This process is depicted in Figure 2.14 and is explained next.

1. First the stored helper data is read from the NVM (w_in).
2. The PUF is challenged, and the helper data is decoded (through the XOR) to form a noisy codeword.
3. The repetition decoder corrects the first flipped bits.
4. The Golay decoder corrects the remaining bits.
5. The hash function is applied after decoding in order to retrieve the cryptographic key on which an attacker has no knowledge even given the helper data.

The Reconstruction procedure is repeated every time data has to be encrypted or decrypted. This is because from a security perspective it is best to remove the key from the system when the system is idle, to make the chance of leaking the key as small as possible.
2.8 Reliability requirements for a Fuzzy Extractor

The previous sections described in detail the work flow of a Secure System based on a PUF. After manufacturing of a secure system, it has to be tested. This section evaluates the requirements a test solution for a Fuzzy Extractor has to fulfill in order to prevent hacking. Hence two sets of requirements have to be considered; the ones related to the quality and the ones related to the security. They are given next.

2.8.1 Test requirements

The test requirements are those that deal with the testability of the device; i.e., number of detected faults. The following test requirements are set for the Fuzzy Extractor:

1. The test solution has to achieve a fault coverage as high as possible. The fault coverage is the ratio between the total number of possible faults in the circuit and the number of faults the test can detect.

2. Testing of the PUF circuitry (e.g. SRAM) is not taken into consideration. The PUF circuitry has its own internal test method and is therefore assumed to be already tested.

3. The developed solution has to be compatible with the industry standard test equipment. Test equipment in the factory is very expensive. It is therefore not feasible to replace it with a custom solution. Small modifications to the test sequence are possible but should be avoided whenever possible.
2.9. SUMMARY

2.8.2 Security requirements

Although a test solution is investigated, security degradation of the device is prohibited. Regarding the Fuzzy Extractor security, the following requirements have to be respected:

1. Refering to Figure 2.11, id_in, id_out and puf_in shall not be revealed at any time, partially or fully. These are the connections where the cryptographic key or the device specific patterns (PUF response) are transmitted in an unencrypted form. This means that an attacker gaining this information can directly use it to his will and the device is not secure any more.

2. Helper data (w_in and w_out in Figure 2.11) are assumed to be public knowledge and do not have to be protected. As an attacker cannot use this data to reconstruct the cryptographic key without the PUF response, this data is considered harmless and leaking it is not a concern.

3. Reverse engineering the Fuzzy Extractor is not an issue. The Fuzzy Extractor makes use of algorithms which are standardized and publicly know. Therefore the functionality of the circuit does not have to be protected.

4. The enrol pin is from a security standpoint a vulnerable pin in the given design, as attackers can potentially repeat the Enrollment phase with a new key and unlock the device. There exist already solutions that solve this issue and therefore the protection of this pin is not considered in this thesis.

The next chapter investigates test methods that fulfill the defined test and security requirements.

2.9 Summary

This chapter reviewed the behaviour of a secure system. It explained what Physical Unclonable Functions are and classified them in two categories: Explicitly-Introduced Randomness PUFs and Intrinsic Randomness PUFs. It continued by discussing their cons and pros and focused on the SRAM PUF type as this one has gained the most popularity (although the methods explained in this thesis are general to any PUF type). Every PUF has one major problem, being the fact that some bits are unstable, i.e. noise. To counter the noise a device called Fuzzy Extractor is used; it performs error correction to protect a key against the noise. The error correcting schemes used in this thesis are: repetition codes and the Golay codes. The combination of them is used to get an optimal error correction capability for a given number of bits. The chapter ends with defining the test and security requirements for the Fuzzy Extractor.
State of the art in secure IC testing

Chapter 2 described a PUF-based secure system and focused on a Fuzzy Extractor and its test requirements. This chapter investigates possible test solutions that meet these requirements. Each approach is analyzed from both a testing and a security perspective. Section 3.1 introduces the basic principles of the test concept. Section 3.2 provides a classification of the different test development approaches and discusses their advantages and disadvantages. Section 3.3 explains the different test application approaches and discusses the combinations that can be made to create an optimal test solution. It also includes a comparison of these methods from a security standpoint. Section 3.4 investigates the improvement that can be made to existing test methods to enhance their security. Section 3.5 summarizes the presented solutions and draws a conclusion.

3.1 Test concept

The manufacturing of electronic devices is a complex process. In highly sophisticated factories, billions of transistors are created on surfaces not larger than a few square millimeters [34, 35]. Due to this, not all these devices are functioning correctly [36]. To separate the faulty devices from the non-faulty ones, they have to be tested. To explain the testing concept, a small example is analyzed. Consider a washing machine that only has a single program to wash the clothes; see Figure 3.1. To test the machine, a test scenario has to be designed. An example of a test scenario is: 1) insert the dirty clothes, fill the machine with detergent and press the start button, 2) wait until the program is finished and 3) check the end result against a known good result (in this example: a clean shirt). Testing of an electric circuits requires a similar testing procedure. A schematic

Figure 3.1: A washing machine

23
overview is depicted in Figure 3.2. The steps are as follows: 1) test inputs have to be generated and applied to the circuit, 2) the values are processed by the circuit and 3) the results are compared against the calculated expected results. If the results are equal the circuit is considered to be defect free; however, if one or more of the comparison fails; the circuit is assumed to have a defect. Test solutions are evaluated according to three metrics:

- **Test time**: amount of time it takes to perform the test.
- **Fault coverage**: the number of faults the test can detect in a device.
- **Cost, such as area overhead**: this is the size of the extra circuit that has to be added to the original circuit, to facilitate its testing.

When developing a test solution two choices have to be made: A *test development approach* and a *test application approach*. The first is described in the coming section and the later in the next section.

### 3.2 Functional vs. structural testing

Test development approaches can be classified into three types: Functional, Structural or a combination of these two types; see Figure 3.3 for an overview. These approaches are explained in more detail in the next subsections.

#### 3.2.1 Functional testing

Functional testing, or black box testing, is a test approach where the design of a device is not considered in developing a test solution; only the function of the device matters, not the structure of the design. Functional testing can further be classified into two classes: Exhaustive testing and Non-exhaustive testing.

**Exhaustive testing** is a method where all possible input combinations are evaluated.
Although these tests are easy to design and implement, the test time is very long for medium to large sized circuits. Take for example an adder that adds two 32 bit values. This circuit consists of 64 inputs and 33 outputs. To completely test this adder, $2^{64} = 18446744073709551616$ input combinations have to be evaluated. Assuming the test can be performed at 1 GHz, this will take $2^{64} \times 10^{-6}/(3600 \times 24 \times 365) = 599.730$ years. This example clearly illustrates that exhaustive testing is only possible for small circuits.

Non-exhaustive testing is a method where a subset of all possible input combinations are evaluated. This reduces the test time, but it cannot guarantee that all faulty devices can be detected.

### 3.2.2 Structural testing

Structural testing, or white box testing, takes the structure of the design of the circuit into account when developing a test solution. It is performed according to a predefined method, which can be automated, and is then called Automated Test Pattern Generation (ATPG). An example is a method where a certain fault in the circuit is modeled and input vectors targeting this fault are derived. By using ATPG, the number of input patterns that have to be evaluated can be reduced significantly. The previous discussed adder can for example be tested by 8 carefully chosen input vectors [19]. If the circuit is tested at 1 GHz the test time would become $8 \times 10^{-6} = 8$ micro seconds. This is a reduction of $10^{18}$ compared to functional testing discussed previously. Several different ATPG algorithms exist. An overview can be found in [37].

Although these algorithms drastically speed-up test time, most circuits still cannot be tested in a reasonable test time, by using only the existing in and output ports. Extra access ports to the internals of the circuit have to be created. This can be accomplished with the following methods:

1. **Test point insertion:** A method of enhancing the testing process through the insertion of extra wires to access difficult to reach nodes in the circuit.
2. **Scan chains:** A method that makes all memory elements in the design accessible for testing purpose.

These two methods are discussed next.

### 3.2.2.1 Test point insertion

To avoid glitches in the circuit, some designs incorporate redundant hardware [38]. Functional testing cannot detect faults that are located in these redundant blocks. In addition, some nodes in the circuit may be hard/impossible to test. A solution is to make these nodes easy accessible e.g. by adding extra wires. This method is called *Test Point Insertion*. An example of this method is depicted in Figure 3.4. In Figure 3.4(a), the output

![Original circuit](image1)

![Circuit with inserted test point](image2)

**Figure 3.4: Test point insertion**

Y can be either zero or one; both values have to be tested to guarantee the correct functioning. To set the output to one, all inputs X1 to X6 have to be set to one. The occurrence probability of this pattern is \( \frac{1}{2^6} \); hence the fault is hard to test. To overcome this problem, a test point is inserted as is depicted in Figure 3.4(b). In the enhanced situation \( x_4, x_5, x_6 \) and the control point are set to one in order to test Y being one; the occurrence probability of the required pattern is now \( \frac{1}{32} \). Due to the higher probability, the AND gate is easier to test in the enhanced situation.

### 3.2.2.2 Scan chains

The other method of enhancing the testing process is the use of *Scan Chains*. An example is depicted in Figure 3.5. The figure contains three flip-flops. During the design phase the flip-flops in the circuit are enhanced with a multiplexor and a wire to a neighboring flip-flop. These enhanced flip-flops are called *Scan Flip-Flops* (SFF). When a test is performed, these scan flip-flops can be connected serially. A series of connected SFFs is called a *Scan Chain*. Through the scan chain test data can be transported to difficult to reach locations inside the circuit. A test sequence consists of three steps: 1) the test data is loaded through the scan chain and the primary inputs into the circuit, 2) the data is processed by the circuit and 3) the results are captured with the scan chain and the primary outputs of the circuit and compared against the calculated correct results.
3.3. ATE VS BIST

Figure 3.5: Overview of a scan chain [39]

3.2.3 Combination

Complex circuit can result in a low fault coverage. Test point insertion or scan chains can alleviate this problem, but might result in a large area overhead. In these situations an optimal test solution could be gained by combining the functional and structural approach. An example of a test scenario where combining functional and structural methods is beneficial is a circuit were exhaustive testing is used for parts of the circuit that are easy to test and structural testing to the remaining parts of the circuit.

3.3 ATE vs BIST

As discussed in Section 3.1, when developing a test solution, a test application approach has also to be covered. There are three types:

1. **External**: *Automated Test Equipment* (ATE). The test patterns are applied by an external machine (ATE) at the input of the device under test and the responses are collected and transferred to the same machine.

2. **Internal**: *Built in self-test* (BIST). The test infrastructure is fully integrated on the device. In the test mode the chip reports whether it is functioning correctly or not.

3. **Combination**: An optimal test strategy could require a combination of the two previous types. For example, generating the test patterns on the chip, while checking the output results with a test machine.

These methods are discussed in more detail next.
3.3.1 External (ATE) testing

*Automated Test Equipment* (ATE) are machines that apply externally test patterns to *Devices Under Test* (DUT), catches their responses and analysis their results. ATEs are specified in the number of outputs they have and the clock rate at which they can apply the test patterns to the device.

The advantages of an ATE are the low area overhead on the device due to the fact that the input patterns and output responses do not have to be stored on the chip. An additional advantage is that extra patterns can be applied to find the exact location of the fault; this is called *diagnosis*. This is often required to improve the production process inside the factory. The disadvantage of an ATE is that the diagnosis capabilities can be misused by hackers to steal secret keys or sensitive data. Another disadvantage is that some devices have to be tested at a lower clock frequency than during normal operation, which increases the test costs.

3.3.2 Internal (BIST) testing

*Built in self-test* (BIST) is a method where the device can test itself without the use of an external test machine. It is a common method for secure devices as this method does not require additional ports that can give access to the internals of the IC and leak any secrets. A schematic overview of a BIST is depicted in Figure 3.6. In the figure, two components are added to the system: a *Test Pattern Generator* (TPG) and an *Output Response Analyzer* (ORA). The TPG is responsible for the input pattern generation and the ORA is responsible for checking the correctness of the output results. Some devices do not need an ORA, due to the fact that the design contains identical components. These components can be tested in parallel by the same patterns; hence comparing their responses provide pass/fail information.

An advantage of BIST is that it can perform the test at the clock speed of the circuit instead of the clock speed of the ATE, thereby reducing the test time. Another advantage is that for secure devices they do not leak any secrets. BIST has two major disadvantage:
Most BIST methods provide a pass/fail as result, which prevents an effective diagnosis. Another disadvantage is that BIST requires some additional design effort.

3.3.3 Combination

There are two main approaches for testing a device: ATE and BIST. However, for complex designs, the test time with an ATE may be considered too long, while a BIST solution might not provide enough diagnosis. In this situation a combination of an ATE and BIST can be used. This can for example be done by partitioning the chip into parts that are tested by an ATE and BIST separately.

3.3.4 Comparison from a security point of view

The previous sections discussed the advantages of ATE and BIST. An overview is depicted in Table 3.1. Test solutions using an ATE result in a very small area overhead and good diagnosis. In addition, due to the fact that tools exist to automatically develop the test patterns, the design effort is small. The main drawback of this method is that hackers might use the diagnosis capability to get access to the internals of the device and secret data can easily leak. BIST has a better security but has limited diagnosis capability and requires a bigger design effort.

Due to the fact that testing with an ATE results in a smaller area overhead and a better diagnosis capability, they are the preferred test solution. However, they are insecure. Solutions to improve their security are investigated in the next section.

3.4 Protecting Secure ICs after manufacturing

The classic approach to get a high quality test of an IC means maximum controllability and observability. This means it should be easy to set a node in the circuit and easy to readout the node at the output. Secure systems require hiding as much as possible detail and access of the internal nodes. Hence there is a trade-off between testability and security. To protect a secure IC after manufacturing, the testing infrastructure should be protected. Protection schemes can be divided into two classes [40], as depicted in Figure 3.7:

- Protection at the I/O ports: The security of an in/output port can be enhanced by introducing a test controller that guards the access to the testing features of
the device. This controller regulates the test sequence in such a way that secrets do not leak.

- **Protection within the system**: System level protection methods add smart schemes within the system to improve the overall security.

![Diagram of test port protecting methods](image)

**Figure 3.7**: Overview of test port protecting methods

The two classes are explained in more detail in the next sections.

### 3.4.1 Protection at the I/O ports

Every device that contains an external test port has two modes of operation: test mode and functional mode [41]. In test mode the test pins are used to load test data into the circuit or read out the test results. In functional mode the circuit is performing its normal functionality. **Protection at the I/O ports** is a class of methods, where the switch from the test mode to functional mode (and/or the other way round), is being protected. The two most well-known methods are explained next.

**Authenticate the user**

This method uses a form of authentication to control the switch between functional mode and test mode. An authorized person provides the test system with a key to unlock the test controller and get access to the internals of the device. A recent proposal, is to use a challenge-response based system to authenticate the user [42]. The authentication process is based on the KATAN block cypher [43]. An overview of this method is depicted in Figure 3.8. It functions as follows: first a random value is generated in the Device Under Test (DUT). This random value is distributed to both the test machine and the hardware KATAN block cipher on the DUT. The test machine calculates by means of a KATAN block cypher an unlocking code based on the random value and a stored key. The same process is performed on the device. The unlocking codes are compared: if they match, the test infrastructure on the chip is unlocked; otherwise it remains locked. Furthermore, an on-chip random generator is needed which increases area considerably.
3.4. PROTECTING SECURE ICS AFTER MANUFACTURING

The advantage of this method is that the secret key to unlock the test infrastructure is encrypted when transferred to the device. The key can thus not be retrieved by probing the wires from the test machine to the DUT. The disadvantages of this method is that for protecting the access a key has to be stored in the chip, which is what PUF based systems try to avoid.

**Removing sensitive information in test mode**

Another method of protecting the test mode is by removing all secret information when switching from functional mode to test mode [44]. An overview of this method is depicted in Figure 3.9. The figure depicts two modes of operation: 1) functional mode and 2) test mode.

![Diagram of removing sensitive information in test mode](image)

**Figure 3.9: Removing sensitive information when switching to system mode**

To change modes, the circuit is forced to pass through reset (mode 3). In the reset state, all the information in the flip-flops is removed and the access to the cryptographic key is blocked.

The advantage of this method is that no extra keys have to be stored on chip and
that standard test machines can be used without any significant modifications. The disadvantage of this method is that it must be guaranteed that during the switch from normal mode to test mode all memory elements are empty. This can cause a large area overhead due to the fact that all flip-flops have to be resettable; or a special circuit has to be designed to empty the flip-flops. In the likely event that an attacker circumvents this procedure, all secret data is accessible.

3.4.2 Protection within the system

The other class of security measures is the protection within the system. These schemes are based on integrated smart techniques within the system to prevent hacking. Five common methods are Fuses, Scan chain integrity, Detecting shifts, Scan-chain scrambling and Obfuscating the scan chain. These are described next.

**Fuses**

The first protection method is to create Fuses in the system. Fuses are small wires that can be destroyed by applying a high voltage. These fuses are located in the wires that connect the test pins to the internals of the device and are destroyed after the test procedure has finished.

This measure can be effective but has two main drawbacks: first, after manufacturing (and testing) some devices might fail in the field. If these devices are returned to the factory, the reason that caused them to fail cannot be easily identified as the test pins are no longer available. Second, there are methods to "repair" the destroyed fuses [5], hence, circumventing this protection method.

**Scan-chain integrity**

When a device contains a scan-chain, a Scan-Chain integrity method can be used to improve security [45]. Shifts in a scan chain are only allowed if the chip is in test mode [46]. If shifts are occurring while the test controller is in functional mode an intrusion into the system is occurring. In a circuit, the scan enable pin is connected from the test controller to the flip-flops via a tree structure. An example can be seen in Figure 3.10. The figure depicts a tree with nine branches. under normal circumstances the branches at the last level of the tree (i.e., at the end of the tree) should have the same logic value. Assume an attacker can probe a wire in one of the sub trees; it can change the logic value of the wire and shift out secret information. The proposed method prevents this by comparing the different branches of the tree: if they are equal the system functions normal; if not, an intrusion is detected and as a counter measure the circuit is reset.

The advantage of this method is its easy implementation. The disadvantage is that it might cost a large amounts of area overhead, due to long wires to compare the different parts of the chip. Moreover, if the clock tree can be probed at the top, an attack is not detected.
3.4. PROTECTING SECURE ICS AFTER MANUFACTURING

Figure 3.10: Scan enable protection [45]

Detecting shifts

As explained in the previous method; shifts in a scan chain are only allowed if the chip is in test mode [46]. The detecting shifts methods aims at detecting illegal shifts in the scans chain by inserting ”spy” flip flops [46]. An overview can be seen in Figure 3.11. In the figure the light (blue) rectangles indicate the normal flip-flops and the dark

Figure 3.11: Spy flip-flops in the scan chain [46]

(red) rectangles indicate the spy flip-flops. The spy flip-flops are not part of the normal functional network of the chip and have a predetermined preloaded value. Assume an attacker is probing the scan enable wire and shifts the flip-flops to retrieve secret information. The spy flip-flops get an opposite value compared to their preloaded value. The
opposite value is detected at the output of the spy flip-flop and the system is signaled that an illegal shift has occurred.

The advantage of this method is its low design effort and easy implementation. The disadvantage is in test mode an attack based on probing is not detected.

**Scan-chain scrambling**

*Scan-chain scrambling* is a method that does not aim at avoiding or detecting an intrusion; instead, it makes the output unusable for an attacker by encrypting the data \[47\]. An overview of this method is depicted in Figure 3.12. In the figure it can be seen that the scan chain is divided into segments. The segments are connected to a *Segments Interconnect Network* (SIN). The SIN connects the scan chains according to the mode of the system; if the system is in test mode the scan chains are connected in a linear order, equal to a normal scan chain. However, if the system is in functional mode the order of the scan chains is determined by a random number generator. This randomly connected scan chain is difficult to decipher making the output data unusable for an attacker.

The advantage of this method is that it is easy to implement. However, the proposed method has major drawbacks: scan chains are placed wide spread over the chip. Long wires are thus necessary to connect the individual scan chains to the muxes and the random number generator. These long wires can be costly in terms of area and can affect the speed at which the test can be performed. Another issue is that a random number generator has to be built on the chip which can be difficult to be get truly random.

**Obfuscating the scan chain**

As with the previous method, methods to *Obfuscating the scan-chain* do not prevent nor detect an intrusion into the system, but instead aim at making the output data unusable by encrypting the data. The encrypting is implemented by adding extra elements to the scan-chain. There are different ways to implement such a scheme:

- **Inserting inverters into the scan chain**: consists of trying to mislead a potential hacker by inserting an inverter in between some arbitrary elements in the scan
chain [48]. An overview of this method can be seen in Figure 3.13. In the figure, four flip-flops are depicted. In between flip-flops 1, 2 and 3, 4 inverters are inserted to mislead potential attackers. Because the attacker does not know their location he does not know the correct value of the flip-flop in the circuit. However, locating the inverters is relative easy with the following method. Assume an attacker knows the circuit; first, in test mode zeros are loaded into the scan-chain. Second, the circuit is switched to functional mode. Third, the output can then be analyzed: at locations were a one is encountered a good estimation can be made where the inverters are located inside the scan chain.

- **Inserting OR elements in the scan chain**: circumvents the easy detection of inverters in the scan chain by inserting OR gates [19]. As the OR function of two zero’s results in a zero, the discovery method of scanning in all zero’s cannot be applied. Despite being more difficult to reverse engineer, this method is still considered easy to circumvent.

The above methods have one thing in common: they are all based on "security by obscurity". This means that their security is based on the assumption that a hacker does not know the layout of the circuit which is considered a bad practice in the security community.

All discussed ATE protection methods have serious security vulnerabilities. As security is a primary concern in testing a Fuzzy Extractor, the options described in this chapter are not further investigated. The next chapter explores the possibilities of using a BIST as the main test method.

### 3.5 Summary

This chapter introduced the test concept and classified the methods according to a test development approach (functional, structural and combination) and test application approach (ATE, BIST and combination). Comparing ATE with BIST, it is concluded that ATE has several advantages, such as less area overhead and better diagnosis capabilities, but a critical disadvantage regarding security. Several methods of protecting ATE where discussed. As all of these methods presented have one or more security vulnerabilities, ATE is no longer considered a viable test solution for a Fuzzy Extractor. The next chapter is dedicated to BIST methods and its implementation.
Chapter 3 classified the test methods in two categories: external and internal test methods. Although external methods had several advantages, it was found that these solutions are not secure enough for use in a PUF based systems. This chapter therefore focuses on an internal test strategy: Built-in self-test. This test method is highly secure, due to the fact that it contains no connections to the outside of the system. Section 4.1 introduces the general concept of Built-in self-test (BIST). One of the main components in a BIST is a test pattern generator. This component will be discussed in Section 4.2. Test pattern generation can be optimized for a specific circuit. Different optimization methods are discussed in Section 4.3. Section 4.4 contains a summary of this chapter.

4.1 Built-in self test concept

_Built-in self test_ (BIST) is a test method that integrates the test infrastructure on the device. A schematic overview of a BIST can be found in Figure 4.1. It works as follows:

![Figure 4.1: Components of a Built-in self-test](image)

1) Input test patterns are generated by the Test Pattern Generator and applied to the circuit. 2) The input patterns are processed by the circuit. 3) The output is compressed with a compactor, to reduce the comparisons necessary. 4) The output of the circuit is compared against the correct results; if they are equal, the circuit is correct, if they are unequal circuit contains a fault. To implement a BIST, four components are required on the device:

1. **Test Pattern Generation**: component responsible for generating the test vectors applied to the circuit under test.
2. **Compaction (optional):** Usually the number of output results of a circuit are large. To reduce the size of the output, a compression/compaction method can be used.

3. **Comparator:** after the test vectors have been applied, the output results have to be compared against the correct results. Thus a comparator is necessary.

4. **Correct results:** The correct results have to be stored on the device in, e.g., a ROM. The correct results are called the *golden reference*.

The two main components in a BIST are the test pattern generation and compaction. These will be discussed in more detail in the coming subsections.

### 4.1.1 Test Pattern Generation

To test a device, input vectors need to be applied to the circuit. In BIST, a component is needed to generate the input vectors on the device. This is called **Test Pattern Generation** (TPG). The on-chip TPG methods can be classified into three categories:

1. **Deterministic:** Input patterns are derived from an ATPG tool and stored in a memory on the device. If the test process is activated the input patterns are applied to the device. The advantage of this method is the short test time. The drawback is the large area due to the storage of the input patterns.

2. **(Pseudo) Random:** A special circuit is designed to generate (pseudo) random numbers. These numbers are used as inputs for the circuit under test. The main advantage of this method is the small area overhead: No patterns have to be stored. The disadvantage is the long test time. It might take a long time before the (pseudo) random number generator outputs the test vector (input pattern) that detects a certain fault.

3. **Combination:** The number of faults a (pseudo) random number generator can detect is often not high enough (within a certain amount of time), while the deterministic patterns cost too much area. Therefore it can be opted to use a combination of random generated and deterministic patterns.

Test pattern generation requires a trade-off between quality and the area overhead. A high quality test, requires a large number of vectors to be stored on chip which uses a lot of area, while a random pattern generator uses a small area, but generates some input vectors that do not detect any new faults and thus reduce the quality.

### 4.1.2 Compaction

To verify the correct functioning of the device, the output results have to be compared with the correct result, known as the *golden reference*. BIST stores the correct results on the device. However, storing the complete correct output result often requires a very
high area overhead. Therefore, compaction (compression) methods are used to reduce the output size. This can be achieved in the following ways [37]:

- **Parity bit**: With this method, from the calculated correct results only the parity bit is stored (the parity bit represents whether the number of ones in a number is odd or even). During test mode from the output of the circuit the parity bit is calculated and compared against the calculated correct result. If the results match, the circuit is considered to be correct, if the results are unequal the circuit is considered to contain a defect. However, this method has one major drawback: a fault may flip multiple bits in the output and thereby preventing a fault from being detected.

- **One counting**: This method stores the number of ones from the calculated correct result. As with the previous method the ones in the output are compared against the correct result stored. If the two results are equal the circuit is considered to be correct. The chance of not detecting a fault is lower compared to the previous method, but requires more area and (because of the calculation of the number of ones) is slower.

- **Multiple Input Signature Register (MISR)**: is a block that calculates for every circuit a unique smaller signature. For a good circuit the signature the block generates can be calculated in software. This signature is stored on the device. During the test process, a comparison is performed, between the stored signature and the signature of the output result of the circuit. If they are the same the circuit is assumed to be correct. If the MISR calculates a different signature the circuit is assumed to contain a defect. The advantage of using a MISR signature is the chance of not detecting a fault is very low compared to previous methods. In addition, it uses a small amount of area and consumes little power.

As can be seen a trade-off has to be made between the number of bits stored and the number of collision that can be accepted (circuits that pass the test while the circuit contain a fault). Due to the robustness against collisions and the small area overhead, the MISR approach is considered most optimal.

Both for test pattern generation and compaction optimizations can be made. However, due to the fact that this thesis focuses on a high quality test with a low area overhead, the focus will be on test pattern generation. This is therefore discussed in more detail in the coming section.

### 4.2 Test Pattern Generation

The most common approach found in BIST designs for test pattern generation is to use a *Pseudo Random Number Generator*. This component implements an algorithm to approximate a sequence of random numbers. To evaluate the different implementations a set of requirements has to be met. These requirements are:
1. **Area overhead:** The test structures on a device are not used for normal functionality and are thus considered overhead. It is therefore a requirement to keep the area as small as possible.

2. **Randomness properties:** The test time of a device should be as low as possible. It is proven that these goals can be met best by introducing a pattern generator that has good random properties [49].

3. **Reconfigurability:** The solution should be easy to tune in order to improve the overall fault coverage.

The state-of-the-art test pattern generation methods are discussed in the next section and are evaluated according to the mentioned criteria.

### 4.2.1 Linear feedback shift register

A **Linear Feedback Shift Register** (LFSR) is a shift register where the next state depends on a linear combination of the previous state. An example of an LFSR is given in Figure 4.2 The blue rectangles indicate a flip-flop connected to the feedback network, the white rectangles indicate flip-flops that are solely connected to their neighbor. The blue flip-flops are connected to XOR elements to calculate the next value for the first flip-flop in the LFSR. The feedback network of an LFSR can be described by a **Characteristic Polynomial**. The general form is described in the following format [50]:

\[
f(x) = 1 + h_1x + h_2x^2 + \ldots + h_nx^n
\]

Where \( h_i = 0 \) or \( 1 \) depending on whether a certain feedback path \( i \) exists and \( n \) the number of flip-flops. For example the Characteristic Polynomial of the LFSR in Figure 4.2 is:

\[
f(x) = 1 + x^{11} + x^{13} + x^{14} + x^{16}
\]

The characteristic polynomials can be divided into two categories:

1. **Primitive:** Primitive polynomials are polynomials that are irreducible. It is proven that an LFSR with a primitive polynomial generates all possible input combinations (except for all zero) [51]. They are therefore called maximum-length
LFSRs. They are used for exhaustive testing. For every size LFSR, there exists exactly one primitive polynomial. An overview of the primitive polynomials for the first eight LFSRs can be seen in Table 4.1.

Table 4.1: Primitive polynomials for the first 8 LFSRs

<table>
<thead>
<tr>
<th>Width</th>
<th>Polynomial</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$x^2 + x + 1$</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>$x^3 + x^2 + 1$</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>$x^4 + x^3 + 1$</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>$x^5 + x^3 + 1$</td>
<td>31</td>
</tr>
<tr>
<td>6</td>
<td>$x^6 + x^5 + 1$</td>
<td>63</td>
</tr>
<tr>
<td>7</td>
<td>$x^7 + x^6 + 1$</td>
<td>127</td>
</tr>
<tr>
<td>8</td>
<td>$x^8 + x^6 + x^5 + x^4 + 1$</td>
<td>255</td>
</tr>
</tbody>
</table>

2. **Non Primitive**: Non primitive polynomials do not have to generate all possible input combinations. A certain fault in the circuit is not detected if it requires an input pattern that is not generated by the LFSR. It is therefore not guaranteed that all faults in the circuit can be detected.

For most BIST implementations the primitive polynomial is chosen as it guarantees that all input combinations are generated. However, it does not necessarily have to be the feedback network that generates the required patterns in the shortest time.

The implementation of an LFSR can be categorized, according to the layout of the feedback network in the following way:

- **External LFSRs**: have the XOR in the feedback path as can be seen in the LFSR of Figure 4.2.
- **Internal LFSRs**: have the XOR elements in between the flip-flops.

The internal LFSR can operate at a higher clock speed due to the fact that the longest path is one XOR delay. Internal feedback LFSRs are therefore favored over external types. LFSRs have a simple structure and are easy to implement. In addition they are easy to optimize. However, they have two major weaknesses:

- **Not truly random**: The sequence produced by an LFSR is high correlation due to the fact that most output values are a shifted version of the previous one. The implication is, that in general, it takes more clock cycles then necessary before an LFSR has reached the required input patterns to achieve high fault coverage.
- **Long wires**: LFSRs have relative long wires as there is a connection from the last element to the first one. This might become an issue with wider LFSRs, as it reduces the speed at which they can operate and complicates the layout of the design.
4.2.2 Cellular automata

To tackle the weaknesses of LFSRs a method was proposed in the form of cellular automata [52]. An overview of this method can be seen in Figure 4.3. In the figure it can be seen that the wires are only connected to the neighboring cells and thus relatively short. Cellular automata are build out of two types of cells, which are named after their feedback pattern; they are:

- **Rule 90**: A cell is connected to its preceding cell and to the next cell. It can be described with the following formula where \( x \) indicates the flip-flop and \( t \) the position:
  \[
  x_i(t + 1) = x_{i-1}(t) + x_{i+1}(t)
  \]

- **Rule 150**: The cell is connected to its previous cell, the current cell and the next cell. It can be described with the following formula:
  \[
  x_i(t + 1) = x_{i-1}(t) + x_i(t) + x_{i+1}(t)
  \]

The even cells in the figure demonstrate the rule 150 feedback pattern, the uneven rule 90.

It is proven that an alternating layout of 90 and 150 cells generates all possible input combinations. A major drawback of Cellular Automata when compared to an LFSR, is that they require more wires and XOR gates, hence more area overhead. In addition they are difficult to optimize for a certain circuit.

4.2.3 Generalized LFSR

Another approach to increase the randomness of an LFSR is defined as a generalized linear feedback shift register (GLFSR) [53]. An example of a GLFSR is depicted in Figure 4.4. In the figure it can be seen that every bit position does not contain one, but multiple values (in this example: two). The number of bits per position can be generalized to \( n \times x \). The different types of GLFSRs are denoted by the format \( (n, y) \), where \( n \) is the number of elements and \( y \) is the number of bits per element. An ordinary LFSR is a special case of a GLFSR and is written as \( (1, y) \).

It has been proven that a GLFSR outperforms both a normal LFSRs and Cellular Automata in terms of the randomness [53]. In Figure 4.5 the randomness of different GLFSRs is shown. The first graph in the figure depicts a standard LFSR as discussed
4.2. TEST PATTERN GENERATION

4.2.1 LFSR

in subsection 4.1 (which is a special case of a GLFSR). In the graph it can be observed that the output has a high correlation in stages that do not have any XOR gate in between the elements. The second picture depicts a GLFSR of type (2, 25), which has a significantly reduced correlation. The third GLFSR (3, 17.5) improves even further. In [53] it is claimed that in general a GLFSR with three bits per element gives the most optimal results. The benefit of a GLFSR with more than one layer of feedback is the good randomness properties it has. The drawback of this method is the complex layout and wiring. In addition, it is difficult to tune or optimize for a specific circuit.

4.2.4 LFSR + phase shifter

A different approach from the previous discussed proposals is to, instead of replacing an LFSR, to enhance it. The enhancement can be achieved by introducing a Phase shifter [54]. The general structure is depicted in Figure 4.6. The phase shifter in the figure is composed of an array of XOR gates. These XOR gates remove the correlations between the channels. A mathematical algorithm is used to calculate where the XOR gates have to be introduced and how much correlation is reduced.

The advantage of this method is that it has a low area overhead compared to the other methods. The drawback of this method is that it cannot always remove the correlation completely and due to the irregular structure is difficult to integrate and slow.
CHAPTER 4. BUILT-IN SELF-TEST

4.2.5 Cryptographic cores

Most secure devices contain an on-chip cryptographic core [55]. Two well know algorithm implemented in a crypt core are the Advanced Encryption Standard (AES) and the Data Encryption Standard (DES). In [55] a method was proposed to self-test these cryptographic cores and thereby eliminating complex addition components to test these devices. In addition, in [55] a method is proposed to use the crypto cores as test pattern generators. The pattern generation is possible due to the good randomness properties of the crypto cores [55]. A schematic overview of an AES or DES core can be seen in Figure 4.7 (except for the blue areas). The main component in the figure is the round module.

It works as follows: 1) first a plain text is initialized. 2) After the initialization, the plain text enters the round module. This module implements the actual encryption scheme.
3) The result is stored in the R1 register. 4) The next clock cycle the result is passed through the feedback wire back to the round module. 5) Optionally, modifications can be made to the circuit, as depicted in blue in the figure, such that the device can be used as an output compaction block. This looping procedure is performed several times (for AES 10 times, for DES 16 times). These rounds are necessary to create the diffusion process; the process which is responsible for the effect that a change in just one input bit creates a big difference at the output. This diffusion process attributes to the good randomness properties of the crypto core.

The advantage of this method is the low area overhead. However, the proposed method has two major drawbacks: first, care should be taken that the testing does not lower the security of the device and second, this method is hard to optimize for a specific circuit.

4.2.6 Summary

Table 4.2 presents an overview of the previously discussed methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Area overhead</th>
<th>Randomness properties</th>
<th>Reconfigurability</th>
</tr>
</thead>
<tbody>
<tr>
<td>LFSR</td>
<td>++</td>
<td>-</td>
<td>++</td>
</tr>
<tr>
<td>Cellular Automata</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Generalized LFSR</td>
<td>+</td>
<td>++</td>
<td>-</td>
</tr>
<tr>
<td>LFSR + Phase shifter</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>AES / DES</td>
<td>++</td>
<td>++</td>
<td>-</td>
</tr>
</tbody>
</table>

From the table it can be concluded that AES or DES components give the best results in both area and randomness. Unfortunately our design does not contain an AES or DES core and thus this method cannot be used. From the remaining methods the LFSR is the smallest in area and the easiest to optimize for a specific circuit. Therefore the remaining part of this chapter focuses on an LFSR, although the optimizations presented are generic and can be applied to the other methods as well.

4.3 Reconfigurable LFSR for improving fault coverage

The faults in a circuit can be split in two categories:

1. **Easily-detected faults**: are faults that can be detected with different input vectors.

2. **Hard to detect faults**: or random-pattern-resistant faults are faults that need a very specific input pattern to be detected.
CHAPTER 4. BUILT-IN SELF-TEST

The random-pattern-resistant increase the test time due to the fact that it may take long for an LFSR to reach the specific input value that detects the fault. To counter this issue two methods exist:

1. **Optimizing the circuit:** enhancing the testing process by making changes to the circuit to remove hard to detect faults. This can be done according to the two methods discussed in Chapter 3: Test point insertion or the introduction of Scan-chains.

2. **Reconfigurable LFSR:** making changes to the LFSR with the aim of generating the desired pattern in an acceptable test time.

Reconfigurable LFSR is discussed in more detail in the rest of this chapter. In this thesis a novel classification is proposed for a reconfigurable LFSR. A motivational example is depicted in Figure 4.8(a). In the figure it can be seen that an LFSR can be made reconfigurable in three locations:

(i) **Output reconfiguration:** in this case the output values of the LFSR are tuned to get the desired input patterns.

(ii) **State reconfiguration:** in this case the values in the flip-flops are changed to get the desired input patterns.

(iii) **Structure reconfiguration:** in this case the feedback network is changed to get the desired input patterns.

An overview of the classification methods can be seen in Figure 4.8(b). These methods are discussed further in the following paragraphs.

**4.3.1 Output reconfiguration**

Output reconfiguration changes the output of the LFSR to get the required input pattern. A schematic overview of this method can be seen in Figure 4.9. In the figure, the
4.3. RECONFIGURABLE LFSR FOR IMPROVING FAULT COVERAGE

white bars indicate the input vectors that detect new faults and the grey bars indicate vectors that do not detect any new faults. Assuming that after vector 14 all faults are detected, the test sequence takes 14 clock cycles. Assume there exists a method that can "map" vectors on other vectors. If vector 14 can be mapped on vector 5 (which previously did not detect any faults), the test length can be shortened. The two most common methods to move vectors are Bit-flipping/fixing and Reconfigurable output. These are discussed next.

**Bit-flipping/fixing**

*Bit-flipping/fixing* are methods where in between the output and the circuit under test components are added [56][57]. An overview of these method can be seen in Figure 4.10. Figure 4.10(a) depicts the Bit-flipping method. At the input of the circuit XOR gates are integrated. Via these XOR gate a bit can be flipped from zero to one and vice-versa. For example, assume the test-pattern to detect a certain fault is 1001 and the pattern currently generated by the LFSR is 1000. If an OR gate is integrated in the circuit at the last output line the bit can be set from 0 to 1 to get the required pattern. Figure 4.10(b) depicts the bit-fixing method. This method integrates AND and OR gates to set the input to zero or one respectively.

The advantage of the bit-flipping/fixing method is that it is easy to implement. The disadvantage is that it moves one vector at a time. If multiple vectors are moved the hardware overhead becomes substantially.
Reconfigurable network

Reconfigurable network is a method where bits in the output of the LFSR can be rerouted to a new location to create the desired input pattern [58]. An overview of the components necessary to implement this method can be seen in Figure Figure 4.11(a); the

![Figure 4.11: Reconfigurable Interconnection Network [58]](image)

figure depicts a system where a Reconfigurable Interconnection Network (RIN) is added in between the LFSR and the circuit. Assume again the pattern needed is 1001 to detect a certain fault, but the pattern generated by the LFSR is 1010. The RIN is now activated to reroute the third bit to the fourth bit and retrieve the desired input vector. Figure 4.11(b) depicts a detailed view of the re-routing network.

An advantage of this method is that its easy to implement. A disadvantage of this method is that it is not guaranteed that all input vectors can be generated, if not enough zeros and ones are available. An example is an LFSR that generates the input combination 1000 and the vector needed is 1001. There are not enough ones in the bit-string to create the desired input vector.

4.3.2 State reconfiguration

State reconfiguration targets the value of the flip-flops inside the LFSR. The concept of this method is depicted in Figure 4.12. In the figure, white bars indicate input vectors

![Figure 4.12: The concept of state methods](image)
that detect faults, grey bars input vectors that do not detect new faults. The concept of state methods is to skip patterns that do not detect new faults by, at certain points in time, reload the flip-flops of an LFSR with a new value. This value is called a "seed". To implement this method, seeds need to be stored on the chip. Three different methods exist:

- **Full reseeding**: the complete internal state of an LFSR is reloaded.
- **Partial reseeding**: a part of the internal state of an LFSR is reloaded, the other flip-flops keep their original value.
- **Built-in reseeding**: the LFSR is reloaded with a pattern that is an encoded version of the current state.

These methods are explained in more detail in the coming subsections:

**Full reseeding**

*Full reseeding* is a method that replaces the value of all flip-flops inside an LFSR. A schematic overview of this method can be seen in Figure 4.13. In the figure it can be seen that the device is enhanced with a memory where patterns are stored.

![Figure 4.13: Full reseeding](image)

The advantage of full reseed is that it very flexible due to the fact that at any time a new pattern can be loaded regardless of the current state of the LFSR. The disadvantage is that the area overhead is high because full seeds need to be stored.

**Partial reseeding**

Instead of storing the complete seed a method was proposed to store only a part of the seed [59]. This can for example be a half or a quarter. The remaining flip-flops keep their current state. An overview of this method can be seen in Figure 4.14. In the figure it can be seen that the memory consists of two parts: a part (seed m), which stores the partial seeds of the LFSR, and a part (n bits) which indicates the part of the LFSR to
be replaced. For example, assume an LFSR of which the seed is split in two parts. An extra bit is necessary to indicate whether the upper half or the lower half of the LFSR needs to be replaced.

The advantage of the Partial reseeding method is the savings in area compared to the full seed storage method. The main drawback of this method is that it is less flexible than full reseeding: The LFSR needs to have the right bits in a part of the LFSR before reseeding can take place.

**Built-in reseeding**

_Built-in reseeding_ is a method where, instead of storing the seeds, the seeds are generated from the LFSR itself by means of logic gates [60]. An overview can be seen in Figure 4.15. In the figure an extra circuit is depicted in the feedback network. The circuit changes the wiring in between the flip-flops, thereby inverting values when needed, to get the required seed.

The main advantage of this method is that the area overhead is small, due to the fact that
4.3. RECONFIGURABLE LFSR FOR IMPROVING FAULT COVERAGE

no seeds are stored in a memory. The main drawback is that it is difficult to implement.

4.3.3 Structure reconfiguration

Structure reconfiguration targets the structure of an LFSR, thereby change the sequence that an LFSR generates. An example is given in Figure 4.16. In the figure the initial seed is the same, but the sequence of patterns they generate are different. This can be exploited to generate the required test vectors in a shorter time. A method that implements structural reconfiguration is Multi-Polynomial. It is explained next.

**Multi-Polynomial**

*Multi-polynomial* is a method that dynamically changes the feedback network of an LFSR during the test sequence [61]. An overview of this method can be seen in Figure 4.17. In the figure it can be seen that AND gates are integrated in the feedback network.

Patterns are stored in a memory that turn on and off certain feedback paths: a one turns a feedback path on, a zero turns a feedback path off.

In [61] the author proofs the method is more efficient than storing seeds, although the possibility exists that certain input vectors cannot be generated. The major challenge with this method is to decide what to change the polynomial to and at which point in time.
4.3.4 Combination

In the previous subsections the methods were discussed independent. However, combining previous methods can provide more optimal results. These methods are discussed next.

Multipolynomial plus Reseeding

A more optimal method compared to previous methods is the combination of Multipolynomial and Reseeding [61]. An overview of this method can be seen in Figure 4.18. The memory contains two parts: the left part contains the polynomial, the right part the seeds. At every reseeding, both the seed and the polynomial are changed.

The advantage of this method is that is easier to tune the LFSR to the required patterns. Theoretical it is not as optimal as pure polynomial reseeding, however, it is more optimal than the seed reseeding method. The main drawback remains the complexity of the implementation.

Open opportunities

An overview of what is discussed in literature is given in Table 4.3. In the table a one indicates an explored method, a zero indicates an unexplored method. In the table it can be seen that not all combinations are evaluated in literature: a method that combines

<table>
<thead>
<tr>
<th></th>
<th>output</th>
<th>state</th>
<th>feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>output</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>state</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Feedback</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>combination</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
feedback with output and methods that combine state, structure and output are not explored.

The previous sections discussed several methods to optimize an LFSR. Chapter 6 evaluates the most optimal method to be used in a Fuzzy Extractor.

4.4 Summary

This chapter discussed *Built-in self-test* (BIST). The BIST method consists of a component to generate the input vectors and a component to check the output result. Test vector input generation can be accomplished by several different methods. The methods discussed in this thesis are: LFSR, Cellular automata, a GLFSR, an LFSR + phase shifter and an AES or DES core. The component considered most favorable for input generation is an LFSR due to its low area overhead and easy tunability. Several methods to optimize an LFSR, to achieve a faster test time, exist. They could be categorized in methods that change the output, the state and the structure.
Chapter 4 introduced BIST and the different optimization schemes that can be used for it. This chapter explores which BIST solution is best for testing a Fuzzy Extractor (FE). Section 5.1 discusses the functional model of the Fuzzy Extractor. Section 5.2 investigates the communication between the FE blocks. Sections 5.3-5.7 explain the FE main blocks (i.e. Golay Encoder, Repetition Encoder, Repetition Decoder, Golay Decoder and Hash Function) and analyzes it from a testing perspective. Section 5.8 analyzes the remaining components. Section 5.9 discusses the testing solution for the complete FE and Section 5.10 summarizes the main contributions of this chapter.

5.1 Fuzzy Extractor functional model

As discussed in Chapter 2, the example Fuzzy Extractor FE under investigation consist of five blocks. A detailed overview, can be seen in Figure 5.1. The technical details of the blocks in the circuit are as follows:

![Fuzzy Extractor Diagram](image)

Figure 5.1: Fuzzy Extractor with the different bit widths between blocks
1. Golay Encoder: accounts for 6.5% of the area in the FE and requires two clock cycles to process data. It contains a 12 bit input bus and a 24 bits output bus.

2. Repetition Encoder: accounts for 7.3% of the area in the FE and requires 267 clock cycles to process the data. It contains a 24 bits input bus and a serial output line.

3. Repetition Decoder: accounts for 6.5% of the area in the FE and requires 290 clock cycles to process the data. It contains a serial input line and a 24 bits output bus.

4. Golay Decoder: accounts for 61.5% of the area in the FE and requires in between 4 and 10 clock cycles to process the data. It contains a 24 input bus and a 12 bits output bus.

5. Hash Function: accounts for 18.2% of the area in the FE and requires 32 clock cycles to process the data. It contains a 12 bits input bus and a 12 bits output bus.

A detailed discussion of the blocks and their communication is given in the coming sections.

5.2 Communication within the Fuzzy Extractor

The blocks inside a Fuzzy Extractor communicate via data buses and control signals. An overview of the data buses in between blocks and their bus width is depicted in Figure 5.1. For example, the bus width between the Golay Encoder and the Repetition Encoder is 24 bits. Given a certain data bus width, three options exist to introduce an LFSR. Every option has its tradeoff between test time and area overhead. The options are:

1. **Smaller than bus width**: an LFSR with a smaller number of outputs than the block bus width can be used if some signals are replicated. Although saving in the number of flip-flops are made, the randomness is reduced significantly thereby increasing the test time. In addition, the solution requires extra wiring which tend to diminish the savings in area.

2. **Matching bus width**: this method is the easiest to implement and optimize. The number of LFSR outputs is equal to the number on inputs of the component to test, but uses more flip-flops than the smaller than bus width LFSR.

3. **Wider than bus width**: an LFSR with more outputs than the block bus width can improve the randomness of the LFSR. However, it costs more area and a proper designed LFSR can achieve similar results.

The LFSR with a number of outputs equal to the block bus width is the best to optimize and therefore the implementation used in this thesis.
In addition to the data buses, each block contains communication signals to handle the communication between blocks. An example is the communication between the Golay Encoder and the Repetition encoder as depicted in Figure 5.2(a). The $out_d$ line transports the data between the Golay Encoder and the Repetition Encoder, the $out_or$ and $in_ir$ lines are the control signals. The control signals implement a handshake protocol which is depicted in Figure 5.2(b). The protocol is necessary due to the fact that each block uses a different amount of clock cycles to process the data. The amount of clock cycles necessary to process the data is defined as latency. The handshake protocol works as follows:

1. The Repetition Encoder signals that it is ready to receive new input by setting the $in_ir$ line to high.
2. Once the data is ready, the Golay Encoder sets its data on the data $out$ bus.
3. To signal that the data is ready the Golay Encoder sets the $out_or$ high.
4. The Repetition Encoder takes the data from the data bus and sets its $in_ir$ bus low. It starts processing the data.
5. The Golay Encoder responds by setting the $out_or$ signal low and is ready to start encoding a new set of data.

Besides the data and control signals, all blocks inside Fuzzy Extractor contain additional signals for proper functioning. These signals are: clock and reset. The clock and reset signal are tested by the automated test machine (ATE), due to the fact that these are required for applying the test.

Our goal is to develop a test method for a Fuzzy Extractor targeting the stuck-at-fault model without using a scan-chain. The patterns are generated by an RTPG and applied functionally (i.e., by using only the input ports). The next sections analyzes the blocks of a FE from a functional point of view and analyzes their testability.
5.3 Golay Encoder

The Fuzzy Extractor consists of five blocks (see Chapter 2). The Golay Encoder is the first block in the Enrollment phase, hence encoding path of the circuit. Its purpose is to make data more robust by adding 12 parity bits to the input data. A detailed explanation of Golay can be found in [62]. Figure 5.3(a) depicts a simplified example of the Golay encoding process with four bits. In the figure a four bit input value is depicted. A one enables a value in the B vector (solid lines in the figure), a zero turns it off, (dashed lines in the figure). The enabled values are Xor-ed to form the parity bits. The output is created by concatenating the input value with the parity bits.

From a testing perspective the most complex part are the value’s to be XOR-ed; This is due to the fact that the hardware created by the compiler to implement this functionality is in the form of an XOR-tree. A simplified example is depicted in Figure 5.3(b). In the figure, the top row indicates the values to be XOR-ed. First the even and uneven values are XOR-ed to produce an intermediate result at the second line in the tree. Second, the intermediate results are XOR-ed to produce the final result. To test this block it must be assured that all bits in the three receive a ’1′ and a ’0′ value at least once during the test. The XOR tree implemented in the Golay Encoder processes 12 values of 12 bits wide. The tree generated contains 4 layers of intermediate values which can be tested by an LFSR in a low number of clock cycles. This is easy to achieve by an RTPG and therefore this block is considered easy to test.

5.4 Repetition Encoder

The Repetition Encoder is the second block in the Enrollment phase, hence of the encoding path. The purpose of this block is to add extra robustness to the Golay encoded vectors. The component contains a 24 bits input bus and a serial output. Every input bit is repeated n clock cycles on the output line. The main part of the block comprises two counters: a source counter and a repetition counter. Figure 5.4 depicts a simplified example of the Repetition Encoder. The figure depicts two counters, Repetition Counter and Source Counter, and a Shift Register. First, the input data is loaded into the Shift
5.5. REPETITION DECODER

Second, the Repetition Counter starts counting the number of times the first bit has to be repeated. Once the correct number of repetitions is reached, it signals the Source Counter to increment. The Source counter increments by one and instructs the Shift Register to shift the bits one position to the left, thereby encoding the next bit. The process is repeated until all bits are encoded. In this example, every input bit is repeated three times. The input value of 1100 becomes thus: 111.111.000.000.

From a testing perspective, the two main components to test are the counters. However, these counters increment regardless of the input value of the circuit, thereby largely testing themselves. The remaining component to test is the shift register Shift Register. It is tested by providing at least a zero and one in every bit position, which is not difficult to achieve with an RTPG. This block is therefore considered easy to test.

5.5 Repetition Decoder

The Repetition Decoder is the first block in the Reconstruction phase, hence decoding path of the circuit. Its purpose is to correct errors in the input stream. It is implemented by counting the number of ’1’s in the input: if there are more than the repetition length divided by two, this bit should be corrected to one, otherwise to zero. The component contains a serial input line and a 24 bits output bus. A simplified example of the repetition decoding process can be seen in Figure 5.5. In the figure, three counters are depicted:

a : the One Counter counts the number of ones in the input stream

b : the Repetition Counter counts until a serie of n bits has passed and subsequently resets the One Counter and increases the Destination Counter.

c : the Destination Counter inserts the bit in the right position of the Output Result.

Assume the input ’101.111.000.100’ is processed and the repetition encoding used in the system is three. The One Counter counts the number of ’1’s in ’101 which is two. The end of the string is notified by the Repetition Counter and the analyzing procedure is started: two is larger than the repetition used in the system divided by two and thus the
CHAPTER 5. TEST DEVELOPMENT FOR A FUZZY EXTRACTOR

Figure 5.5: Motivational example of a Repetition Decoder

first decoded bit decoded is '1'. The '1' is written at the first position in the Output Result as indicated by the Destination Counter. The Destination Counter is incremented to point to the next bit in the Output Result to which the next decoded bit is written. This procedure is repeated until the complete input is decoded.

From a testing perspective the main components are the One Counter, Repetition Counter and Destination Counter. The Repetition Counter and Destination Counter are self-incrementing, thereby testing themselves. The One Counter is tested by supplying the input with a string of ones which is easy to achieve by an RTPG. The remaining parts (i.e. some comparison circuit) are not hard to test either. Due to this the block is considered easy to test.

5.6 Golay Decoder

The second block in the Reconstruction phase, hence decoding path, is the Golay Decoder. It is the most complex block in the circuit, due to the fact that it contains a large state machine for decoding the input vectors. A schematic overview of the state machine is depicted in Figure 5.6. In the figure nine states are depicted. The states 1, 8 and 9 are states that solely copy data between registers. The states 2 and 5 do some straightforward calculations; these are therefore not discussed in detail. The most complex states are: state 3, 4, 6 and 7. Analyzing the Golay Decoder from the testing perspective, the following conclusions can be drawn:

1. The most complex states are state 4 and state 7
2. As depicted in Figure 5.6 the state machine requires specific input vectors to reach those states. I.e., vectors that do not contain faults will skip all intermediate states.

To test the Golay Decoder, each case in the state-machine needs to be activated and checked. Thus, using a generic RTPG for this block can also result in high fault coverage
5.7 Hash Function

The last component in the decoding path of the Fuzzy Extractor is the hash function. Its purpose is to increase the entropy of the bits at the output of the system. The component contains a 12 bits input and a 12 bits output bus. A simplified example of the hash function is depicted Figure 5.7. The figure depicts three components: an Input Buffer an LFSR and an Accumulator. First the input data is loaded into the Input Buffer. The Input Buffer represent the s_input in the source code. The Accumulator holds the final result and represent the accu_reg in the source code. The Input Buffer is analyzed bit per bit: if a bit is one, the LFSR is added (Xor-ed) to the Accumulator, if
the bit it zero the bit is skipped. Assuming the input '1100' is processed, the LFSR is 
Xor-ed two times with the Accumulator to produce the output result.

Analyzing this block from the testing perspective reveals that the most complex part 
is at line 4-9. However, this block is easy to test, because the LFSR defined on those 
lines shifts regardless of the input value, thereby testing itself. The block is therefore 
considered easy to test with a RTPG.

5.8 Remaining components

The Fuzzy Extractor contains besides the blocks previously discussed additional compo-
nents; e.g., muxes, XOR gates etc, with the purpose of connecting the Fuzzy Extractor 
to the outside world (i.e., memory, co-processors, etc). These components are easy to 
test, and can be tested by an integrating test on a higher test hierarchy and are therefore 
not the main focus of this thesis.

Analyzing the different blocks reveals that the Golay Decoder is the most difficult block 
to test. This is due to the fact that it contains states that are hard to test and the state 
machine requires very specific test patterns to reach those states.

5.9 Test methods and procedure

This thesis proposes two different methods to test a Fuzzy Extractor; they are scan-chain 
free thereby not degrading the security:

- **Daisy chaining**: is a method where the blocks are tested in a loop-chain, reusing 
  existing blocks in the FE; hence resulting in a low area overhead.
- **Parallel**: a method where the blocks are tested separately and in parallel, reducing 
  the overall test time.

These methods are explained next.

5.9.1 Daisy-chaining

As discussed in Section 5.7, the main component of the Hash Function is an LFSR. This 
LFSR can be used as an input generator and an output compressor for the complete 
circuit. A Fuzzy Extractor with the self-test method is depicted in Figure 5.8. In the 
figure, it can be seen that two new connections are added to the circuit:

1. Connection between the repetition Encoder and the Repetition Decoder. This 
   connection transports vectors from the encoding path to the decoding.
5.9. TEST METHODS AND PROCEDURE

2. Connection between the output and the input of the circuit. This connection transports the random output results of the hash function to the input of the circuit.

When analyzing this method one issue arises: connection 1 transports perfect input vectors (i.e., vectors that do not contain noise from a PUF) from the encoding path to the decoding path of the Fuzzy Extractor. For the Repetition Decoder this is not of issue, however, as discussed in Section 5.6, the Golay Decoder contains several paths in the state machine and the path taken depends on the location of the faults in the code word. Due to the perfect input vectors, three out of four states of the state machine are not tested and thus the result is a low fault coverage. To tackle this problem, a new component is introduced that simulates faults in the input vectors to activate the paths that otherwise would not be tested. This component is the MISR/SISR as discussed in Section 4.1. The MISR/SISR can be introduced at several places in the design. An overview of the possible locations is depicted in Figure 5.9. Analyzing the different options the following three conclusions are made:

1. Option a depicts the reference situation without any test components added to the circuit.

2. Options b and c are similar. This is due to the fact that a Repetition Encoder followed directly to a Repetition Decoder act as a wire (a decoder is the inverse of
3. Option \(d\) results in the smallest LFSR. Another advantage is that the LFSR is not placed between components and thus does not have any effect on the critical path.

4. Options \(e\) and \(f\) might produce slightly better results due to the fact that two LFSRs produce more random results than one. The differences in the test results should be minor as a good designed LFSR should have good randomness properties.

To determine the impact of the different solutions, the next chapter investigates options \(a, b, d\) and \(f\).

### 5.9.2 Parallel

A Fuzzy Extractor with the parallel testing method components is depicted in Figure 5.10.

Figure 5.10(a) shows that each block of the Fuzzy Extractor has a dedicated LFSR or MISR for test pattern generation and a MISR for output result compression. A detailed view of the MISR circuitry is depicted in Figure 5.10(b). As an example, the test sequence for the Golay Encoder is as follows:
5.10 Summary

This chapter analyzed the detailed functionality of each of the five blocks comprising the Fuzzy Extractor. Additionally, it discussed the suitability of the blocks for being tested by means of a random pattern generator. Thereafter two different secure testing methods are proposed, the daisy-chaining method and the parallel method. The first testing method connects the FE blocks in a loop-chain, reusing the hash function block of the FE as both a random pattern generator and as an output compressor to minimize the area overhead. Due to the characteristics of the Golay Encoder / Decoder pair, connecting the blocks in a loop-chain introduces a new challenge. Additional hardware needs to be introduced in between these two blocks to increase the fault coverage, e.g., MISR/SISR. Five constructions with possible locations for the new hardware are proposed. The second method tests each of the blocks individual and simultaneously, reducing the overall test time.

1. The LFSR at the input of the Golay Encoder generates input stimuli for the block. The methods to optimize this process are those discussed in Chapter 4.

2. The MISR at the output of the Golay Encoder is configured to compress the output results. At the same time it generates input stimuli for the next block (Repetition Encoder)

This test sequence is applied to all blocks in the Fuzzy Extractor in parallel. If a high fault coverage is reached the MISRs are stopped and compared with the golden reference. If they match, the Fuzzy Extractor is considered fault free. The next chapter investigates the impact of the parallel test solution.
Chapter 5 proposed two secure test methods; daisy-chaining method and parallel method. This chapter describes the experiments to evaluate and compare the proposed test methods, it shows and discusses the results of the experiments and it gives a list of recommendations to test a generic Fuzzy Extractor. Section 6.1 describes the experiments performed on the two proposed test methods. Section 6.2 shows the results of testing the Fuzzy Extractor with the daisy-chaining method. Section 6.3 shows the results of testing the Fuzzy Extractor with the parallel method. Section 6.4 compares and discusses the trade-offs between the two test methods. Section 6.5 gives a list of recommendations to test a generic Fuzzy Extractor. Section 6.6 summarizes the main contributions of this chapter.

6.1 Experiments

Chapter 5.9 proposed two new methods to securely test a Fuzzy Extractor:

1. **Daisy-chaining**: to minimize the area overhead, the Universal Hash block is reused for both input pattern generation and the output compression while the blocks are connected in a loop-chain. Additional hardware is required to increase the randomness between the Golay Encoder / Decoder pair.

2. **Parallel**: to minimize the overall test time, the blocks are tested individual and simultaneously by additional random test pattern generators; these are used for both input pattern generation and the output compression.

To evaluate and compare the proposed test methods, regarding their fault coverage, test time and area overhead, experiments are performed. They are described next.

6.1.1 Daisy-chaining

To determine the best configuration proposed in the previous chapter, see Figure 5.9, in terms of highest fault coverage with shortest test time, six experiments are perform:

1. **Default**: in this experiment, the circuit is tested with the blocks connected in a loop (as depicted in Figure 5.9(a)) for $15 \times 10^4$ clock cycles and the fault coverage is analyzed. The test time of this experiment is used as test time budget for all the experiments.
2. **MISR**: in this experiment, the circuit is tested configured in a loop with a MISR in between the Repetition Decoder and the Golay Decoder to improve the randomness (as depicted in Figure 5.9(b) and 5.9(c)).

3. **SISR**: in this experiment, the circuit is tested configured in a loop with a SISR in between the Repetition Encoder to the Repetition Decoder to improve the randomness (as depicted in Figure 5.9(d)).

4. **SISR + MISR**: in this experiment, the circuit is tested in a loop with both a SISR and a MISR added to the circuit to improve the randomness (as depicted in Figure 5.9(e) and 5.9(f)).

5. **Default + SISR**: in this experiment, the circuit is tested in two stages. First, as in Figure 5.9(a), the FE is tested using the default loop-chain for 25% of the test time budget. Second, as in Figure 5.9(d), the SISR is included in the loop (between the Repetition Encoder and Repetition Decoder blocks) and the fault coverage is analyzed over the remaining 75% of the time. The goal of this experiment is to analyze the impact of combining the default scenario in Figure 5.9(a) with the SISR in Figure 5.9(d) on the fault coverage.

6. **Default + MISR**: in this experiment, previous procedure is repeated, but in the second stage a MISR is added instead of a SISR (a combination of the methods depicted in Figures 5.9(a) and (b)).

### 6.1.2 Parallel

To determine the impact that the state (seed) and structure (polynomial) have on the fault coverage and test time, nine experiments are performed per FE block; they are described next. Note that conceptually, output methods and state methods are very similar, hence, the influence of the output methods can be easily derived from the results of the experiments carried out for the state and structure methods.

1. **State**: in this experiment, each FE sub-block is tested with a random test pattern generator (primitive polynomial) using three different initial starting conditions or seeds; these are:
   (i) the ‘0...00001’ seed, i.e., all bits are zero except the last bit, which is a one.
   (ii) The ‘10...0110’ seed, a randomly chosen starting state and
   (iii) the ‘01...10101’ seed, a string of alternating zeros and ones.

2. **Structure**: in this experiment, each FE sub-block is tested with a random test pattern generator using three different polynomials; these are: primitive polynomial, and two randomly chosen non-primitive polynomials. The size of the polynomials depends on the number of input bits of the block being tested. All polynomials use ‘0...00001’ seed, i.e., all bits are zero except the last bit, which is a one.
3. **State + Structure:** in this experiment, the combined impact of varying both the seed and polynomial is tested; i.e., each FE sub-block is tested with a random test pattern generator using for each of the three polynomials defined above each of the three initial conditions (or seeds) defined previously.

The program used to analyze the fault coverage is LIFTING [63] developed by Laboratoire d’Informatique de Robotique et de Microelectronique de Montpellier (LIRMM). LIFTING is an open source fault simulator optimized for functional BIST.

### 6.2 Test results of daisy-chaining

This section shows the results of the six performed experiments defined in previous section regarding the daisy-chaining test method. Figure 6.1 shows the fault coverage (y-axis) versus number of clock cycles (x-axis) of the six experiments. Part (a) of the figure shows the first four experiments, while part (b) shows the remainder experiments that include the default stage.

*Figure 6.1: Fault coverage versus clock cycles for Fuzzy Extractor tested with daisy-chaining method*

From Figure 6.1(a) the following can be observed:

1. During the default stage the Fuzzy Extractor realizes a fault coverage of only 36%. This fault coverage is realized quickly in the first 2k clock cycles. The figure clearly shows that the fault coverage remains stable at 36% during the remaining clock cycles.

2. For the remaining schemes, after $5 \times 10^4$ clock cycles the fault coverage reaches 95.1%. The remaining $10 \times 10^4$ clock cycles lead to an increment of only 1.2% (from 95.1% up to 96.2%) fault coverage.

3. There seems to be no major difference in fault coverage between SISR, MISR and their combination. All the techniques realize the same fault coverage in similar test time, i.e., 95.1% is achieved at $5 \times 10^4$ (50k) clock cycles.

From Figure 6.1(b) the following can be observed:
1. Switching to SISR/MISR after the default stage strongly increases the fault coverage, i.e., the fault coverage increases from 36% up to 95.1% in $6.4 \times 10^4$ clock cycles.

2. After switching to SISR/MISR, it takes $5.8 \times 10^4$ clock cycles to reach a fault coverage of 95.1%. Extending the test time to $15 \times 10^4$ increases the fault coverage to 96.3%.

3. The impact of the two experiments on the fault coverage seems to be marginal, i.e., the impact of combining the default stage with either a SISR or MISR on the fault coverage is negligible.

The area overhead of the different configurations is measured in 0.35μm technology node; the results are given in Table 6.1.

<table>
<thead>
<tr>
<th>Area overhead of Daisy-chaining different configurations</th>
<th>Area ($\mu m^2$)</th>
<th>Area overhead (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>$2.96 \times 10^5$</td>
<td>0%</td>
</tr>
<tr>
<td>MISR</td>
<td>$3.17 \times 10^5$</td>
<td>6.8%</td>
</tr>
<tr>
<td>SISR</td>
<td>$3.03 \times 10^5$</td>
<td>2.2%</td>
</tr>
<tr>
<td>SISR + MISR</td>
<td>$3.20 \times 10^5$</td>
<td>8.0%</td>
</tr>
</tbody>
</table>

The results show that testing the circuit with the default configuration results in the smallest area overhead. However, this method realizes a very low fault coverage, as seen previously. From the remaining configurations, the SISR results in the smallest area overhead, i.e., only 2.2% extra area overhead when compared with the default configuration. Additionally, SISR configuration realizes the same fault coverage, therefore, this method is considered to be the best configuration to test our Fuzzy Extractor with the daisy-chaining method.

### 6.3 Test results of parallel

This section shows the results of the nine performed experiments per block defined in the previous section regarding the parallel test method. The results are discussed block per block.

#### 6.3.1 Golay Encoder

Figure 6.2 shows the fault coverage (y-axis) versus number of clock cycles (x-axis) of the nine experiments. Part (a) of the figure shows the primitive polynomial and the three different initial conditions (or seeds), part (b) shows the first non-primitive polynomial and the three different initial conditions, while part (c) shows the second non-primitive polynomial and the three different initial conditions.
6.3. TEST RESULTS OF PARALLEL

The figure shows that for the Golay Encoder, the impact of varying the polynomial and or seed is marginal; i.e., a very high fault coverage is always achieved in a few clock cycles regardless of the polynomial or seed initially chosen. The maximum fault coverage obtained is 98.2% with the primitive polynomial and seed 0 and the minimum is 94.4% with non-primitive polynomial 2 seed 1, a variation of 1.04×. The fastest combination of polynomial and seed to reach first the minimal required fault coverage of 95% is the non-primitive polynomial 2 with seed 0 that achieves a fault coverage of 95.7% after 15 clock cycles.

Moreover, the results reveal that for some polynomial/seed combinations the fault coverage does not increase for certain intervals of time (clock cycles); e.g., Figure 6.2(b) - non-primitive polynomial 1, seed 2, between clock cycles 6 and 11. This particular type of shape of fault coverage is a natural candidate for the optimization techniques described in Chapter 4. These are discussed later in this chapter.

6.3.2 Repetition Encoder

Figure 6.3 shows the impact of different polynomials and different initial seeds on the fault coverage of a Repetition Encoder. For the Repetition Encoder, the impact of varying the polynomial and or seed is significant; e.g., after 1k clock cycles, the primitive polynomial with seed 0 achieves a fault coverage of only 79% while non-primitive polynomial 2 with seed 0 achieves a fault coverage of 90.7%, so an improvement of 1.15×.
6.3.3 Repetition Decoder

Figure 6.4 shows the impact of different polynomials and different initial seeds on the fault coverage of a Repetition Decoder.

For the Repetition Decoder, the impact of varying the polynomial and or seed is significant; e.g., after 600 clock cycles, the non-primitive polynomial 1 with seed 2 achieves a fault coverage of only 82.1% while both primitive polynomial seed 1 and non-primitive polynomial 2 with seed 0 and achieve a fault coverage of 97.1%, so an improvement of $1.18 \times$.

6.3.4 Golay Decoder

Figure 6.5 shows the impact of different polynomials and different initial seeds on the fault coverage of a Golay Decoder.

For the Golay Decoder, the impact of varying the polynomial and or seed is marginal. After 8k clock cycles, the maximum fault coverage obtained is 96.3% with the non-primitive polynomial 2 and seed 2 and the minimum is 94.3% with non-primitive polynomial 2 seed 1, so an improvement of $1.02 \times$. No combination is particularly faster.
6.3.5 Universal Hash

Figure 6.6 shows the impact of different polynomials and different initial seeds on the fault coverage of a Universal Hash.

![Fault coverage versus clock cycles](image)

(a) Primitive polynomial   (b) Non-primitive polynomial 1   (c) Non-primitive polynomial 2

Figure 6.6: Fault coverage versus clock cycles for the Universal Hash tested with parallel method

For the Hash Function, the impact of varying the polynomial and or seed is marginal. After 500 clock cycles, the realized fault coverage is 93.1% for all cases.

6.3.6 Impact of reconfigurable LFSR/RTPG parameters

Section 4.3 reviewed state-of-the-art methods to improve the test quality; these were classified into output methods, state methods and structure methods. In this section, a state method will be applied, the Reseeding method, to improve the test quality, i.e., to anticipate test vectors to decrease the test time, of the Golay Encoder block. As stated previously, in Figure 6.2(a) - primitive polynomial, seed 0, it is evident that, e.g., clock cycles 3, 9 and 10, no faults are detected. Clock cycles that do not contribute to the fault coverage are therefore skipped by means of reseeding, i.e., by changing the seed of the registers of the RTPG. Figure 6.7(b) shows the results of applying the reseeding method to the Golay Encoder block. Part (a) of the figure shows an histogram with the number of faults detected per clock cycle when no reseeding method is applied, while part (b) shows the fault coverage versus clock cycles of testing the Golay Encoder block with the parallel method with and without reseeding method.

The Figure 6.7(a) shows that several intervals of time detect a low number of faults; e.g., clock cycles 9 and 10, no additional faults are detected. Additionally, Figure 6.7(b) shows the impact on the fault coverage and on the test time of skipping the clock cycles that detect either zero or one fault. A speedup of $1.6 \times$ is achieved by reseeding four times, while realizing the same fault coverage. On the downside, the area overhead is very large when compared to the test without reseeding. Each seed requires 12 bits, hence, a total of 48 bits (12 bits × 4 seeds), is stored in memory. Due to the small size of the Golay Encoder (the circuit represents just 6.5% of the FE area), the memory and it’s additional components, e.g., the row and columns decoders, control unit, are estimated to increase the size of the circuit substantially. This extra area overhead makes the optimization methods prohibitively expensive for our case study circuit. In addition, the circuit does not have the longest test time of the FE blocks. Therefore, in the remaining
CHAPTER 6. RESULTS

Figure 6.7: Reseeding results for the Golay Encoder

part of this thesis, this optimization is not considered. Table 6.2 summarizes the results obtained in the previous sections.

Table 6.2: Area and fault coverage per block

<table>
<thead>
<tr>
<th>Block Name</th>
<th>Area (%)</th>
<th>Fault coverage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Golay Encoder</td>
<td>6.5%</td>
<td>99.3%</td>
</tr>
<tr>
<td>Repetition Encoder</td>
<td>7.3%</td>
<td>90.7%</td>
</tr>
<tr>
<td>Repetition Decoder</td>
<td>6.5%</td>
<td>97.1%</td>
</tr>
<tr>
<td>Golay Decoder</td>
<td>61.5%</td>
<td>96.4%</td>
</tr>
<tr>
<td>Universal Hash</td>
<td>18.2%</td>
<td>93.1%</td>
</tr>
</tbody>
</table>

The total fault coverage of the parallel method is calculated with the following formula:

$$\text{total fault coverage} = \sum (\text{block area} \times \text{fault coverage})$$

The realized total fault coverage of parallel method while testing the Fuzzy Extractor is 95.6%.

6.4 Comparison of the test methods

An overview of the two methods discussed in previous sections is depicted in Table 6.3.

Table 6.3: Overview of the results of Fuzzy Extractor testing methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Area ($\mu m^2$)</th>
<th>Area overhead (%)</th>
<th>Test time (CC)</th>
<th>Fault coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Daisy-chaining w/ SISR</td>
<td>$3.03 \times 10^7$</td>
<td>2.2%</td>
<td>50k</td>
<td>95.2%</td>
</tr>
<tr>
<td>Parallel</td>
<td>$3.52 \times 10^7$</td>
<td>18.6%</td>
<td>8k</td>
<td>95.6%</td>
</tr>
</tbody>
</table>

Analyzing the table, a trade-off can be seen between area overhead and test time. For example, testing the Fuzzy Extractor with the daisy chaining method results in an area overhead $9 \times$ smaller, but $6.25 \times$ slower than testing the blocks parallel. On the contrary, if all blocks are tested parallel, the test time is that of the block with the longest test
time: the Golay Decoder with 8k clock cycles. The disadvantage is the much larger area overhead of 18.6%. Between these extremes a more fine trade-off can be made by combining blocks during testing. It is up to the designer to make the most optimal choice for a given design. Please note that an additional advantage of testing the blocks individually is the enhanced diagnosis. A fault can be detected per block. However, diagnosis is not of primary concern in this thesis.

6.5 Recommendations

Fuzzy Extractors might comprise different blocks than the Fuzzy Extractor used as a case-study in this theses; hence, a generic step-by-step procedure is provided for secure testing any FE based on our findings. These steps are:

- The first step is to identify each block and to deeply understand its functionality (which is critical for successful functional testing).

- Second, the characteristics of each block need to be assessed (e.g., its area overhead, if the block comprises a state-machine or not, state-machine complexity, etc).

- Third, perform testability analysis of each block by considering a random input source. Identify eventual challenges by testing a certain block with this method.

- Fourth, implement parallel test method to derive the shortest test time and implement daisy-chain test method to derive the shortest area overhead.

- Fifth, identify if there is need of extra components in order to increase the fault coverage, such as a SISR. If so, analyze the trade-off of such components in terms of its possible locations and of its impact on security, fault coverage, test time and area overhead.

- Sixth, optimize the test solution; optimize parallel test method by choosing an optimal polynomial and seed or optimize Daisy-chain test method by using a SISR or MISR to activate uncovered paths in the state-machine.

- Seventh, analyzed the FC test time and area overhead of all test methods separately and when combined with complementary schemes.

- Finally, determine and select the best test method and complementary schemes to meet the design requirements.

6.6 Summary

This chapter described the experiments to evaluate and compare the daisy-chaining and the parallel test methods. There were performed six experiments for the daisy-chaining method and nine experiments for the parallel test method. It was demonstrated that
both test methods realize a similar fault coverage; however the daisy-chaining method has a smaller area overhead, i.e., 2.2% against 18.6% but a longer test time, i.e., 50k clock cycles against 8k clock cycles. Moreover, to explore the potential of the methods to improve the test quality, the reseeding method was applied on one of the Fuzzy Extractors blocks, the Golay Encoder. The results showed that the same fault coverage was realized in $1.6 \times$ less clock cycles. Finally, a list of recommendations to test a generic Fuzzy Extractor was provided.
Conclusion & Future work

Chapter 6 showed the results of two secure test methods for a Fuzzy Extractor proposed in this thesis. This chapter summarizes the contributions of this thesis and analyzes future work. Section 7.1 summarizes and discusses the conclusions of this thesis. Section 7.2 gives recommendations for further research.

7.1 Contribution

In the context of secure testing a Fuzzy Extractor with functional BIST, this thesis makes the following contributions.

- Two efficient test methods for a Fuzzy Extractor that do not degrade the security of the system: daisy-chaining and parallel. Both methods result in a fault coverage of 95%. Daisy-chaining method reuses existing blocks in the Fuzzy Extractor minimizing the area overhead; it requires 50k clock cycles and has an area overhead of just 2.2%. Parallel method tests the blocks separately and in parallel minimizing the test time to only 8k clock cycles and has an area overhead overhead of 18.6%. Within these boundaries trade-offs between test time and area overhead can be made.

- A new method of input generation and output compression of test results based in a hash function. It is demonstrated that the method results in a high fault coverage of 95% in 50k clock cycles, with an area overhead of 2.2%.

- A new classification of test method optimizations. By using this new classification accurate predictions can be made regarding the best method for a certain application, with its respective trade-offs.

7.2 Future work

Based on the developed work in this project recommendations are given for further research. These recommendations are as follows:

- The proposed test methods presented in this thesis result in a high fault coverage. However, the location of the fault is not indicated. A future research project can investigate methods to extend our test solutions with diagnosis.
• The proposed test solutions do not reach 100% fault coverage. However, there exist designs for which a fault coverage of 100% is required. A future research project can investigate methods to increase the fault coverage.

• The focus of this research was on testing Fuzzy Extractors. However, during the research phase there were clear indications that the methods presented in this thesis could be used to test other circuits. A future research project can investigate which other circuits can benefit from the testing methods developed in this thesis.
Bibliography


[8] [Online]. Available: http://arstechnica.com/old/content/2006/03/6445.ars


[31] [Online]. Available: www.unique-security.eu/


[34] [Online]. Available: http://rel.intersil.com/docs/lexicon/Manufacture.html


[38] [Online]. Available: http://www2.elo.utfsm.cl/~lsb/elo211/aplicaciones/katz/chapter3/chapter03.doc4.html


